UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS DE SEVILLA

Departamento de Ingeniería Electrónica

Contribuciones a la integración de instrumentación electrónica y sistemas embebidos en entornos Web

Memoria presentada por Sergio Gallardo Vázquez para optar al grado de Doctor por la Universidad de Sevilla.

Director: Federico José Barrero García

Sergio Luis Toral Marín

Sevilla, Octubre de 2015

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Los procesos en los que se apoyan la ciencia y tecnología generan variables físicas, las cuales se pueden medir mediante instrumentos que tienen como misión determinar la magnitud de una variable, visualizarla, generarla o convertirla en otra diferente. El proceso de medición requiere el uso de instrumentos como medios físicos para determinar la magnitud de una variable. Los instrumentos, por tanto, constituyen una extensión de las facultades humanas y en muchos casos permiten a las personas determinar el valor de una cantidad desconocida, la cual no podría medirse utilizando solamente las facultades sensoriales, por lo tanto, un instrumento se puede definir como un dispositivo para determinar el valor o la magnitud de una cantidad o variable. El instrumento electrónico, como lo indica su nombre, se basa en principios eléctricos.

En la actualidad, la instrumentación electrónica afronta constantes cambios, convirtiéndose en una herramienta indispensable para ingenieros, científicos y técnicos que requieren de sistemas electrónicos de medida y estimulación de gran exactitud y precisión. Por un lado, el continuo avance de la microelectrónica, las prestaciones de los paquetes informáticos y el desarrollo de nuevas tecnologías en el diseño de sistemas de medida y estimulación para el control de procesos, verificación de productos, explotación de servicios, análisis de calidad, etc., han permitido el desarrollo de potentes sistemas automatizados de medida (ATE, Automated Test Equipment); mientras que por otro lado, es cada vez más común la utilización de computadoras personales (PC, Personal Computer) como principal recurso en diversas áreas de aplicación, tales como laboratorios, entornos industriales y sistemas de instrumentación, entre otros.

Dentro de este contexto, el desarrollo de sistemas de instrumentación remota es una multidisciplina de creciente interés gracias al gran avance que estamos viviendo en el acceso a la información de nuestra sociedad.

El estudio del estado del arte y la técnica de estas tecnologías arrojará que, desde antaño, empresas, organizaciones educativas y entidades de investigación han mostrado un enorme interés por la instrumentación remota; dando solución a numerosos problemas que, sin este tipo de desarrollos difícilmente serían alcanzables.

Distintas empresas de instrumentación del sector se han hecho eco, desde sus albores, de esta necesidad, desarrollando potentes paquetes software que, como veremos a lo largo del presente trabajo de tesis, ofrecen la oportunidad de crear potentes arquitecturas de instrumentación remota que se aproximan, constantemente, a dar solución a los problemas de la sociedad (empresa, educación e investigación, fundamentalmente).

No obstante, gracias al ingente desarrollo de nuevos modelos de programación, la aparición de sistemas multiplataforma, potentes entornos gratuitos de desarrollo y tecnologías orientadas a servicios Web; se plantea la posibilidad de acometer complejas soluciones de instrumentación basándonos

en las mismas; aspecto que será desarrollado a lo largo del presente documento, respondiendo afirmativamente a la anterior cuestión.

Por último, resulta fundamental poder establecer parámetros no absolutos que permitan medir las bondades de los sistemas de instrumentación, más allá de la percepción del desarrollador, en este sentido, la percepción subjetiva del usuario final tiene gran valor, al mismo tiempo que la medición de parámetros objetivos que nos permitan establecer criterios de mejora en este tipos de plataformas.

Todos y cada uno de estos aspectos son los que el presente trabajo de tesis desarrolla a lo largo del documento.

1.1. Objetivos

El interés por el desarrollo de sistemas de instrumentación remota accesibles a través de la Web nos ha conducido a cuestionarnos sobre las bondades y limitaciones que el sector especializado de la instrumentación nos ofrece; al mismo tiempo que estudiar posible mejoras dentro de este ámbito de aplicación.

Con el presente trabajo de tesis se persigue dar respuesta a la cuestión de poder implementar sistemas de instrumentación remota empleando tecnologías no especializadas en sistemas de instrumentación, aprovechando la potencia, versatilidad y flexibilidad que este tipo de tecnologías ofrecen e integrándolas dentro de un nuevo modelo de diseño de sistemas de instrumentación remota accesible vía Web que pueda ser empleado en diversos sectores y aplicaciones.

De este modo, en primer lugar se propondrá el desarrollo de un laboratorio de instrumentación remota basado en una de las más potentes plataformas de desarrollo de aplicaciones de instrumentación virtual existentes en la actualidad; estudiando sus bondades y limitaciones. Seguidamente, y partiendo de las conclusiones obtenidas de la anterior propuesta, se persigue poner de manifiesto la posibilidad de desarrollar sistemas de instrumentación remota accesible vía Web empleando tecnologías abiertas y mejorando las prestaciones que pueden alcanzarse con otras soluciones comerciales. Con ello, los objetivos se concretan en:

- Estudio del estado del arte y técnica del sector de la instrumentación remota.
- Propuesta de una arquitectura de instrumentación remota accesible vía Web empleando plataformas comerciales como punto de referencia.
- Análisis de virtudes y limitaciones de la solución propuesta.
- Propuesta de un nuevo modelo de sistema de instrumentación basada en tecnologías Web abiertas.
- Análisis de virtudes y limitaciones del nuevo modelo y comparativa de prestaciones con el modelo de referencia.

A lo largo del presente documento se desarrollarán y desglosarán los objetivos inicialmente descritos; tratando aspectos como son los relativos a la mejora de interfaces, reducción de costes, aumento de la portabilidad de las soluciones, mejoras de la portabilidad, utilización de modelos de clientes delgados que faciliten el acceso desde diferentes dispositivos o la independencia de tecnologías, entre otros.

1.2. Organización de la tesis

La estructura de la presente tesis comienza en el capítulo 2, donde se desarrolla un breve estudio del arte y la técnica de los sistemas de instrumentación electrónica. El capítulo comienza describiendo el origen y evolución de los sistemas de instrumentación hasta llegar al concepto de "instrumentación virtual". Seguidamente se describen algunos aspectos relacionados con las interfaces de instrumentación que podemos encontrar, para pasar a discutir sobre algunas de las principales aplicaciones software que se emplean y relacionar posibles arquitecturas de acceso remoto que podemos emplear para el diseño de sistemas de instrumentación. Finalmente se describen algunos aspectos sobre el ámbito de aplicación de la instrumentación virtual remota. Seguidamente, el capítulo 3 describe el sistema o bancada de instrumentación que se va a emplear para la implementación de los modelos de sistemas de instrumentación remota propuestos, con objeto de contextualizar algunos aspectos que serán tratados a lo largo del presente documento. El capítulo 4 se relaciona con la descripción y discusión de la primera propuesta de laboratorio de instrumentación remoto desarrollado, en adelante, eDSPlab, basado en la utilización del software LabVIEW. A continuación el capítulo 5 desarrolla una nueva propuesta de sistema de instrumentación virtual basado en tecnología AJAX en el extremo cliente y JAVA en el servidor, al que se ha denominado eDSPlab+, relacionando los principales elementos y tecnologías implicados y discutiendo sobre las soluciones adoptadas. Tras las descripción de los dos laboratorios de instrumentación remota desarrollados, el capítulo 6 propone un conjunto de parámetros y métricas que permitan determinar la bondad de cada propuesta, analizando los resultados asociados al comportamiento de ambos. Finalmente, el capítulo 7 describe las principales conclusiones obtenidas y se presentan comentarios sobre futuros trabajos de investigación basados en este trabajo. El documento de tesis termina con las referencias bibliográficas y anexos.

2. Estado del arte y de la técnica

La instrumentación electrónica podría definirse como una disciplina de la electrónica que versa sobre el diseño y manejo de los aparatos electrónicos en general y, particularmente, los aparatos de excitación y medida. Estos aparatos de excitación y medida son los que denominaremos, en adelante, instrumentos electrónicos, instrumentación electrónica, o simplemente instrumentación.

A su vez, la realización de una medida requiere, en numerosas ocasiones, la intervención de varios instrumentos, unos generan estímulos sobre el dispositivo que se pretende medir y otros recogen la respuesta a estos estímulos. Este conjunto de instrumentos que hace posible la realización de la medida recibe el nombre de sistema de instrumentación. Todo sistema de instrumentación consta, por lo tanto, de unos instrumentos, un sistema de interconexión de estos instrumentos y un controlador inteligente que gestiona el funcionamiento de todo el sistema y da las órdenes para que una medida se realice correctamente [1], [2].

2.1. Evolución de la instrumentación electrónica

La historia de la instrumentación electrónica está íntimamente relacionada con la propia evolución de la tecnología electrónica desde sus orígenes. La inherente necesidad de medir y cuantificar las magnitudes eléctricas impulsó la creación de una disciplina para tal efecto dentro del área de la tecnología electrónica, lo que vino a denominarse Instrumentación Electrónica.

Si bien es cierto que los albores de la electricidad datan de las primeras observaciones de Tales de Mileto al frotar una barra de ámbar con un paño, no será hasta finales del siglo XVIII cuando la electricidad comience a dar sus primeros frutos como disciplina de la investigación y de la ciencia.

Una de las primeras experiencias de la electricidad data del año 1771, cuando el científico Galvani (1737 – 1798) observa que se produce una contracción muscular al poner en contacto un metal con las patas de una rana. Erróneamente, Galvani interpreta este fenómeno como “electricidad animal” y propone que el cerebro de los animales producía electricidad que se transfería a los nervios y, tras ser acumulada en los músculos, llegaba a producir el movimiento de los miembros [3], [4].

Pasarán algunos años para que Alejandro Volta (1745 – 1827) interprete correctamente el experimento de Galvani: la pata de la rana actuaba como un mero “detector de electricidad” y, en realidad, la electricidad era producida por la unión del metal y la disolución que le rodeaba. Esta experiencia inspira a Volta en la invención en el año 1800 de la primera pila, formada por una serie de discos de plata y cinc separados por papel humedecido en salmuera [3], [4].

Curiosamente, el mecanismo de cómo se producía electricidad en la pila de Volta no fue conocido hasta bastantes años después. Sin embargo, ello no impidió que, en 1820, Hans Christian Oesterd (1777 – 1851) realizara un experimento demostrando que el paso de una corriente eléctrica por un

conductor cambiaba la dirección de una aguja magnética cercana al mismo[1]. Este fenómeno causó mucha excitación en la comunidad científica, dado que establecía una conexión entre la electricidad y el magnetismo, lo que dio lugar a la disciplina que hoy conocemos como electromagnetismo [3].

No fue, sin embargo, hasta 1822, cuando André Marie Ampère (1775 – 1836) descubre y desarrolla las leyes que justifican el desvío que sufre de una aguja magnética al paso de una corriente eléctrica, al mismo tiempo que desarrolla el sistema astático, tan usado después en el diseño de galvanómetros, lo que hace posible el funcionamiento de los actuales aparatos de medida. La unidad de intensidad de corriente eléctrica, el amperio, recibe este nombre en su honor [5].

Posteriormente, se suceden varios descubrimientos e invenciones hasta llegar a la invención del primer galvanómetro, ideado por Nobili en 1825. Durante esta época se introducen numerosos modelos diferentes de galvanómetros hasta que, en el año 1881, aparece el galvanómetro actualmente conocido como de Deprez-D'Arsonval o galvanómetro de bobina móvil, diseñado por Jacques D'Arsonval a partir del galvanómetro de Deprez. Un instrumento, verdaderamente útil el cuál, con muy pocas modificaciones, continúa empleándose en la actualidad [5].

La fecha de 1881 adquiere, además, una especial significación para la disciplina de la electricidad, dado que, por primera vez, se dedica la Exposición Universal de Paris de dicho año íntegramente a esta temática [5].

A partir de esta fecha, se enfatiza el interés de la industria por los aparatos de medida, hasta entonces postergados al mundo científico y académico, especialmente los aparatos de medida directa, presentándose, en el año 1884, el primer amperímetro y el primer voltímetro, diseñados por los ingenieros eléctricos británicos Ayrton y Perry [6].

La aparición del voltímetro y del amperímetro vino acompañada de una intensa controversia y polémica acerca de la precisión de sus medidas, polémica que no residía tanto en este aspecto como en las prácticas y la cultura experimental asociadas a los nuevos instrumentos, incluyendo aspectos que abarcaban el campo de la ética y la moral inclusive [7].

Fue precisamente el impulso de la industria de la telegrafía la que propició la adopción del primer patrón de resistencias consensuado a nivel internacional, lo que tuvo lugar en la ya mencionada Exposición de Paris de 1881. Fijado dicho patrón, comienzan a aparecer los primeros aparatos de medición de resistencias, los óhmetros u ohmiómetros, basados en la invención del galvanómetro como el resto de instrumentos [5].

El comienzo del siglo XX marcará un nuevo hito en la historia de la instrumentación electrónica. Este siglo se caracteriza por un creciente interés por la corriente alterna, lo que nos llevará a la aparición de otros de los aparatos de medición que han revolucionado la historia de la instrumentación electrónica: el osciloscopio [5].

[1] Este hecho ya había sido observado en 1.802 por el físico italiano Gian Domenico Romagnosi, pero fue ignorado

La historia del osciloscopio se desarrolla entre varias disciplinas de la Física, como la acústica, la óptica, la electricidad y el magnetismo. A lo largo del siglo XIX se desarrollaron varios métodos basados en la óptica, lo que condujo a la aparición de los primeros estroboscopios en los años 30. Tendremos que esperar a la década de los 80 del mismo siglo para encontrar la primera versión eléctrica del osciloscopio, introducido por Jules Joubert, aunque la captura de una onda comportaba de “cuatro a cinco horas” y su uso fue rápidamente descartado por la industria [8].

No será hasta 1897, de la mano del premio novel de Física Carl Ferdinand Braun, cuando aparezca el primer osciloscopio moderno al adaptar un tubo de rayos catódicos de manera que el chorro de electrones del tubo se dirigiera hacia una pantalla fluorescente por medio de campos magnéticos generados por la corriente alterna [5].

Poco después aparecerán nuevas invenciones como el diodo o válvula de vacío en 1904, de la mano de Fleming, inicialmente ideado para la medición de corrientes alternas de alta frecuencia, y el triodo de Forest en 1906, el primer amplificador [4]. Esto, junto al impulso de los avances científicos durante la Segunda Guerra Mundial, extiende el uso de los modernos osciloscopios como instrumentación fundamental de los laboratorios, al mismo tiempo que la industria de la instrumentación se consolida y se suceden sendos aparatos de medida comerciales por crecientes empresas del sector [5].

Paralelamente, fruto de las anteriores invenciones, y auspiciada por la II Guerra Mundial, se produce una enorme revolución en la industria de la computación, apareciendo lo que se conoce como Primera Generación de la Historia de la Computación, lo que supone la aparición de los primeros computadores [9].

Poco después de finalizar la Segunda Guerra Mundial, en 1947, aparece el transistor semiconductor, ideado en los laboratorios Bell, lo que supuso una nueva revolución en la historia de la electrónica y de la computación [9].

En relación con lo que nos atañe, la instrumentación electrónica, se observa que los sistemas de desviación mecánica utilizados en los aparatos de medida se sustituyen, paulatinamente, por tecnología digital, encontrándonos, en 1952, con el primer voltímetro digital, aún basado en la tecnología de válvulas de vacío, y creado por Andrew Kay, fundador de Non-Linear Systems.

El interés por el procesamiento y almacenamiento de la información proveniente del mundo exterior propicia la integración de la hoy día indisoluble unión de la electrónica y la computación. Uniendo el mundo de la electrónica analógica y la electrónica digital.

En los años sesenta, coincidiendo con la aparición de los primeros circuitos integrados, crece el interés por las redes digitales de transmisión de la información, lo que constituye el terceto formado por electrónica, computación y comunicación.

A mediados de la década de los años 60, en 1965, una consolidada empresa fabricante de equipos de medida, Hewlett-Packard (actualmente Agilent Technologies) desarrolla un bus que permite la interconexión de sus sistemas de instrumentación mediante el uso de una computadora, lo que denominó HP-IB (Hewlett-Packard Instrumentation Bus). El éxito de esta idea fue tan rotundo, que el sector de la instrumentación electrónica comienza a demostrar un creciente interés por dicha implementación, hasta el punto que otros fabricantes copian el HP-IB, llamando a su implementación General Purpose Interface Bus (GPIB o bus de interfaz de propósito general) [10], [11].

No obstante, la falta de un estándar que compatibilizara la interconexión de las computadoras con los sistemas de instrumentación de distintos fabricantes hizo que, pocos años después, en 1975, y partiendo de la implementación inicial de este bus, la oficina de normas del IEEE aprobara la norma IEEE 488-1975: “IEEE Standard Digital Interface for Programmable Instrumentation”, apareciendo, por tanto, el primer bus estándar que permitía comunicaciones entre instrumentos electrónicos y un ordenador. Esta norma será parcialmente modificada en 1978, dando lugar al bus normalizado IEEE 488-1978 [10], [11].

La norma 488-1978, sin embargo, no hacía referencia a la sintaxis ni formatos de los comandos o datos enviados por el bus, por lo que fue necesario modificarla nuevamente, apareciendo así, en 1987 las normas IEEE 488.1-1987 e IEEE 488.2-1987 [10], [11].

La norma IEEE-488.1-1987 renombraría a la original norma IEEE-488-1978 y definiría el nivel físico del bus, esto es, parámetros mecánicos, eléctricos, y el protocolo básico del bus GPIB, sin hacer referencia al formato de los datos o comandos. Por otra parte, la norma IEEE-488.2-1987, “Codes, Formats, Protocols, and Common Commands for IEEE-488.1”, implementada sobre la anterior, proporcionaría el nivel lógico del bus, definiendo la sintaxis básica, los formatos de los mensajes, así como los comandos independientes de dispositivo, estructuras de datos, protocolos de error, y similares [10], [11].

Mientras que IEEE-488.1 definía el hardware, y IEEE-488.2 definía la sintaxis, todavía no existía un estándar para comandos específicos de cada instrumento, lo que implicaba que los comandos para controlar la misma clase de instrumento podían variar entre diferentes fabricantes e incluso modelos [10], [11].

En 1990, esta problemática se resuelve con una propuesta presentada por Hewlett-Packard, el SCPI (Standard Commands for Programable Instrumentation) [10], que especifica un extenso conjunto de comandos estándares en formato ASCII para ser utilizados por todos los fabricantes de equipos de instrumentación programables. SCPI especifica el procedimiento para el intercambio de mensajes, el formato de datos y el modelo de reporte de datos. Nace así el primer lenguaje basado en comandos, independiente del fabricante del equipo, para el manejo de instrumentación programable y se crea un consorcio de empresas fabricantes de instrumentos programables con la misión de velar por el desarrollo de éste [11].

Paralelamente, la aparición de la tecnología HP-IB y GPIB propuestas a mediados de los años 60 impulsan el interés por otras arquitecturas de sistemas de instrumentación, tal es el caso de la tecnología VXI (VME Extensions for Instrumentation), adoptada en 1987, y basada en el bus VME. VXI consistía en una arquitectura de instrumentos modulares implementados en tarjetas que se insertaban en un chasis, dando solución a muchas aplicaciones de sistemas de instrumentación que requerían la integración en el mismo sistema de medida de varios instrumentos, con una velocidad de procesado y transmisión de datos muy grande [12].

Tras la creación de VXI se decidió avanzar un paso más en la abstracción en los ATE (Automated Test Equipment), buscando que la tecnología de interconexión de los sistemas ATE a los PC fuera transparente al desarrollador, independientemente de tratarse de instrumentación GPIB, VXI o cualquier tecnología subyacente [11].

En 1992 la norma IEEE-488.2 sufre una nueva revisión y, un año después, la compañía Nacional Instruments presenta la especificación HS 488 (High Speed 488) destinada a conseguir una velocidad mayor que con la tecnología IEEE-488, compatible con el conjunto de especificaciones del estándar IEEE 488. En el mismo año, IEEE adopto al bus VXI bajo el estándar IEEE 1551 [11].

Fruto del esfuerzo por estandarizar el acceso a los instrumentos de forma transparente a la interfaz física, surge VISA ,Virtual Instruments Software Architecture, un convenio de las empresas Agilent y National Instrument [11].

VISA permitirá comunicarse a través de interfaces de instrumentación como VXI, GPIB y otras del mismo modo, es decir, se convierte en una librería de comunicaciones estándar que permite hacer la tecnología de interconexión del sistema ATE transparente al usuario, pudiendo coexistir varias interfaces de comunicación en el mismo sistema [14], convirtiéndose en una herramienta vital para la interoperabilidad de los componentes del sistema, reduciendo el tiempo y esfuerzo en la programación de los interfaces de instrumentación.

Poco años después, en 1998, surge el consorcio IVI (Interchangeable Virtual Instruments) entre una treintena de compañías, incluyendo usuarios de sistemas como Boeing, y fabricantes de hardware como Agilent, Tektronix, NI, etc, con el objetivo de alcanzar una estandarización de los drivers de los instrumentos para solucionar el problema de la “intercambiabilidad” [14].

IVI se convierte en una herramienta para el desarrollo de drivers de instrumento compatibles, favoreciendo la compatibilidad de una aplicación desarrollada con drivers de instrumento proporcionados por el fabricante, permitiendo la simulación de instrumentos no conectados al sistema y facilitando el desarrollo de drivers a los fabricantes de instrumentos [14].

La tecnología IVI adoptaría VISA, independizando la programación respecto a la interfaz de comunicación. Posibilitaría el intercambio de instrumentos, incluso de distintos fabricantes, pudiendo trabajar con instrumentos simulados durante el desarrollo de aplicaciones. Para ello, los drivers IVI serían clasificados en ocho tipos, correspondientes a fuentes DC, multímetros digitales, generadores de funciones, osciloscopios, medidores de

potencia, generadores de RF, analizadores de espectros y conmutadores de señales (switches) [14].

Con una librería IVI, el programador de un lenguaje de programación podría emplear rutinas de alto nivel sin necesidad de conocer el conjunto de comandos SCPI que el instrumento entiende ni la interfaz de comunicación que éste emplea. Como consecuencia, el desarrollo de aplicaciones de esta forma se agilizó considerablemente respecto al uso de comandos SCPI [14].

Las interfaces de instrumentación de los sistemas ATE cuentan, en la actualidad, con diversas combinaciones de hardware y software que les permiten aumentar su funcionalidad, capacidad de intercambio de datos, facilidad de manejo y operación, etc., con el objetivo de ofrecer una mejora en la conectividad entre sistemas de instrumentación.

2.2. Instrumentación virtual

Fruto de la ingente evolución de la instrumentación electrónica y de su integración con las tecnologías de la computación y la comunicación, surge en 1984, acuñado por la firma National Instruments, el concepto de instrumento virtual [16].

La evolución experimentada por la industria de la computación en los años 80 propiciaría la popularización del uso ordenador, lo que favoreció, a su vez, su inclusión dentro de los sistemas de instrumentación. Esto cambiará la visión de los instrumentos de medición como elementos aislados e independientes, siendo integrados o comunicados con ordenadores, permitiendo enfatizar sus propiedades y dando lugar a una revolución en la instrumentación de ensayos, medición y automatización [17].

En la actualidad, la utilización del ordenador en sistemas de instrumentación permite al usuario manipular un instrumento de medida que no es real, accedido a través del PC, gracias a un determinado software. Éste interactúa con la computadora como si estuviese utilizando un instrumento real. Este instrumento “virtual” se ejecuta en una computadora, tiene sus características definidas por software, se apoya en un hardware de instrumentación, y puede realizar las mismas funciones que un equipo real e incluso ampliarlas [2].

El término "virtual" nacerá, por tanto, a partir del hecho de que cuando se utiliza el PC como "instrumento" es el usuario mismo quién, a través del software, define su “funcionalidad” y "apariencia" y por ello decimos que "virtualizamos" el instrumento, ya que funcionalidad y apariencia pueden ser definidas una y otra vez por el usuario [2].

Consecuentemente, podemos afirmar que un instrumento virtual no es más que un módulo software que simula cada uno de los aspectos funcionales del instrumento real, basándose en todos los dispositivos físicos que pueden ser accesibles para el ordenador (tarjetas de adquisición, tarjetas DSP, instrumentos accesibles vía GPIB, VXI, RS-232, etc.) y haciéndolos accesibles a través de una interfaz de usuario que permite su monitorización y control [2], [12], [18], [19].

La idea es sustituir y ampliar elementos "hardware" por otros "software", para ello se emplea un PC que ejecute un programa específico, éste se comunicará con los instrumentos para configurarlos y leer sus medidas. En muchas ocasiones el usuario final del sistema de instrumentación sólo ve la representación gráfica de los indicadores y botones de control virtuales en la pantalla del ordenador [2], [18].

El instrumento virtual, definido entonces como una capa de software y hardware que se le agrega a un PC, permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida" [18], [17].

El concepto de "instrumentación virtual", por tanto, abarcará más allá de la simple medición, involucrando el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz hombre-máquina, las funciones de análisis y procesamiento de señales, las rutinas de almacenamiento de datos y la comunicación con otros equipos [2], [17].

2.2.1. Instrumentación virtual versus instrumentación convencional

Por una parte, la concepción tradicional de un instrumento de medida se asimila a un dispositivo hardware capaz de capturar señales y proporcionar medidas en una interfaz, generalmente gráfica. Sumándose a lo anterior, existe una creciente tendencia a añadir funcionalidades limitadas de almacenamiento, escalabilidad y comunicación a éstos. Del otro lado, desde la perspectiva de la instrumentación virtual, el ordenador, mediante un hardware de adquisición o un hardware de comunicación conectado a un instrumento de medición, recoge dichas señales de campo y las procesa; por tanto, el ordenador se convierte en el dispositivo clave del sistema de medida y control, al recaer el acento del sistema de instrumentación sobre el software, en lugar del hardware. Precisamente, es gracias a la utilización de las tecnologías de la computación, de donde se derivan la mayoría de las virtudes que podemos atribuir a la instrumentación virtual [2].

El uso del ordenador en los sistemas de instrumentación conlleva ventajas como la facilidad de conectividad con redes de datos, lo que hace posible utilizar un único sistema de instrumentación basado en PC que proporcione medidas a varias computadoras locales o remotas, en las que se ejecuta el código del instrumento virtual [2]. Esto permite múltiples capturas desde un solo punto, con la posibilidad de envío de la información, de forma selectiva, a múltiples puntos locales o remotos, pudiendo realizarse entre distintas plataformas basadas en diferentes sistemas operativos. Esta solución propicia aunar recursos y posibilita entornos de instrumentación colaborativos y multiusuarios [20], [21], [22].

Otra virtud derivada de la inclusión del PC en un sistema de instrumentación es que, debido a la gran capacidad de almacenamiento de éstos, rápido acceso a la información y toma de decisión, brindan la posibilidad de emular una gran cantidad de dispositivos de medición y operar varios instrumentos al mismo tiempo [2].

En los sistemas tradicionales, el instrumento es implementado por el fabricante, mientras que los “instrumentos virtuales” son definidos por el usuario; la interfaz y, en muchas ocasiones, el instrumento propiamente, recaen sobre el diseño del usuario [29]. Esto es así dado que, en un instrumento virtual, el software es la clave del sistema, a diferencia del instrumento tradicional, donde la clave es el hardware. Podríamos, por ejemplo, construir un osciloscopio "personalizado", con la interfaz gráfica que se desee, agregándole inclusive más funcionalidad. Sin embargo, este mismo sistema puede también ser utilizado en la medición de temperatura, o en el control de arranque/parada de una bomba centrífuga. Es aquí donde radica uno de los principales beneficios de la instrumentación virtual, su flexibilidad. Este instrumento virtual no sólo nos permitirá visualizar una onda, sino que a la vez nos permite representar gráficamente su espectro de potencia, valor medio, etc. de forma simultánea [17], [2].

También podemos afirmar que la instrumentación tradicional dispone de funcionalidad y conectividad limitadas, mientras que el uso del PC oportuna una funcionalidad ilimitada, orientada a aplicaciones y con amplias capacidades de conectividad, es decir, existe una mayor escalabilidad [17], [2].

En cuanto a la relación de coste versus funcionalidad implementada, es menor en el caso de la instrumentación virtual al mismo tiempo que es posible aumentar la funcionalidad software y reutilizarla con relativa facilidad [2].

La rápida incorporación a las nuevas tecnologías, gracias a la versatilidad del PC, contrasta frente a la dificultad, cada vez más evidente, de la adaptación de la instrumentación tradicional a éstas, a lo que se une a la alta economía de escala y bajo coste de mantenimiento frente a la instrumentación tradicional.

De forma resumida, la flexibilidad, el bajo coste de mantenimiento, la reusabilidad, la personalización de cada instrumento, la rápida incorporación de nuevas tecnologías, el bajo costo por función, el bajo costo por canal, etc. son algunos de los beneficios que ofrece la instrumentación virtual. Pudiendo también ser implementada en equipos móviles [23], [24], de sobremesa [22], equipos distribuidos en campo [25], equipos a distancia o equipos industriales [26], entre otros.

2.2.2. Justificación de la instrumentación virtual en el mundo científico y empresarial

En la actualidad, podemos encontrar numerosas áreas de la ciencia, la industria y la sociedad que precisan de una amplia y continua cooperación y colaboración internacional. En muchos casos, esta cooperación impera cuando se precisa la accesibilidad a sofisticados equipos de instrumentación que, frecuentemente, es inasequible localmente, bien por su coste o bien por el conocimiento que requiere su utilización. Por tanto, el desarrollo y difusión de arquitecturas, técnicas y tecnologías que propicien acceso virtual, remoto y compartido a los sistemas de instrumentación supone un hito esencial de la Sociedad de la Información y del Conocimiento. La posibilidad de utilizar equipos de instrumentación independientemente de su localización física, contribuye a la equidad y unificación de comunidades industriales y científicas,

abriendo nuevas oportunidades de negocio y conocimiento para la industria, la ciencia y la sociedad. Añadido a lo anterior, el impacto político y estratégico es vital, pues nos lleva hacia una sociedad más unificada [27], [28].

La identificación sistemática de instrumentos y de las correspondientes comunidades usuarias, la definición de sus requerimientos, así como el cuidadoso análisis de la sinergia de la instrumentación remota con redes de comunicación e infraestructura de alta velocidad de la próxima generación, serán la base para definir las recomendaciones para diseñar la próxima generación de Servicios de Instrumentación Remotos [27], [28].

Desde el punto de vista técnico, podemos afirmar que, al no utilizar software y hardware preestablecido, ingenieros y científicos obtienen máxima flexibilidad definida por el usuario. Si bien un instrumento convencional proporciona tanto software como circuitos de medición en un producto con funcionalidad fija utilizando el panel frontal del instrumento. Un instrumento virtual proporciona todo el software y hardware necesario para lograr la medición o tarea de control. Aunado a un instrumento virtual; ingenieros y científicos pueden ajustar la adquisición, análisis, almacenamiento, unión, y funcionalidad de presentación, reutilizando una aplicación en múltiples dispositivos y/o implementando múltiples aplicaciones en un único dispositivo [29].

Sumándose a lo anterior, numerosas empresas del sector de la instrumentación electrónica como, por ejemplo, National Instruments, están enfocando su actividad comercial en la adaptación y/o reutilización de arquitecturas y tecnologías de alta inversión de compañías como Microsoft, Intel, Analog Devices, Xilinx, etc. Apoyándose en la ingente inversión en sistemas operativos y herramientas de desarrollo de terceras compañías, caso de, por ejemplo, Microsoft, y aprovechando la inversión en hardware de conversión de otras como, por ejemplo, Analog Devices [29].

Este creciente interés por los sistemas de instrumentación virtuales es especialmente relevante en la comunidad de científicos e ingenieros que disponen en sus laboratorios una combinación tanto de instrumentos, ya sean los basados en tarjetas de adquisición como tradicionales. La utilización de los buses de comunicación presentes en la mayoría de los instrumentos tradicionales de hoy día son el punto de encuentro hacia la compatibilidad de estos, lo que es posible gracias al software de instrumentación virtual que proporciona los mecanismos necesarios para crear estas interfases de instrumentación [29].

2.3. Interfaces de instrumentación

Un sistema de instrumentación virtual debe implementar un mecanismo que permita separar o intermediar en el control de los dispositivos de la lógica que coordina su operación para obtener los resultados finales, como el control de éste o la representación de datos [30]. Uno de estos mecanismos, conocidos como “middleware” para interfaces de instrumentación, es el estándar SCPI [31].

En los albores de los años 90, un grupo de fabricantes de sistemas de instrumentación anuncia la norma SCPI [16], cuyo objeto es conseguir una

estandarización de los comandos de control y el formato de los datos de los instrumentos, definiendo un conjunto de órdenes comunes para programar sistemas de instrumentación [16], [1]. Se pretende que, independientemente del fabricante, equipos que comparten la misma funcionalidad atiendan de igual forma a un conjunto estándar de comandos [1].

En la actualidad, la norma SCPI constituye el escalón más alto dentro de la jerarquía de normas para el control de sistemas de instrumentación [1]. Ésta se asienta sobre la norma IEEE-488.2 y esta, a su vez, se basa en la IEEE-488.1, no obstante, la norma SCPI puede utilizarse para el control de cualquier tipo de instrumento, independientemente la interfaz o del tipo de bus de comunicaciones que utilice, como son los sistemas basados en VXI, RS-232, USB, Ethernet, etc. [16], [1], [10].

Un último aspecto a considerar es que, el estándar SCPI es algo más abstracto que IVI, no teniendo que tener consideraciones acerca de la red o lenguaje de programación a utilizar. El uso de comandos SCPI estándar lo convierten en la mejor elección frente al uso de IVI, dado que proporciona una representación más unificada y fácilmente extensible de los instrumentos, no dependiendo de ningún protocolo o sistema operativo en concreto [30].

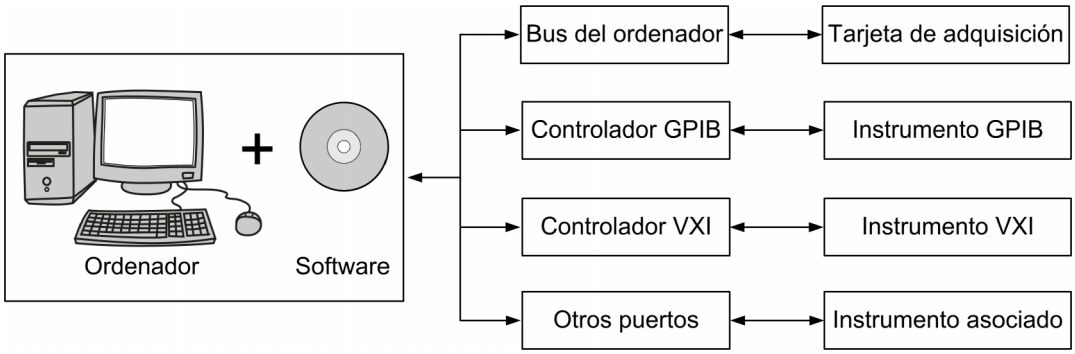


Figura 1.- Diferentes tipos de instrumentos

2.3.1. Tecnologías de instrumentación virtual programable

Existen infinidad de dispositivos electrónicos que podemos considerar sistemas de instrumentación [17], lo que ha dado lugar a un ingente desarrollo de las tecnologías de interconexión entre estos y otros sistemas, naciendo así el primer nivel de la instrumentación virtual: Las interfaces de instrumentación.

Fruto de este amplio desarrollo de las interfaces de instrumentación virtual, algunos autores han propuesto diversas clasificaciones de instrumentación virtual atendiendo a dichas tecnologías de interconexión. Por ejemplo, una primera clasificación podría distinguir 4 tipos de instrumentos: tarjetas de adquisición de datos, instrumentos controlados vía RS-232, instrumentos controlados vía IEEE-488 e instrumentos VXI [12]. No obstante, hoy día esta clasificación podría considerarse en desuso tanto por la integración de tecnologías como por la aparición de otras nuevas [32].

Una clasificación más acertada que nos ayudaría a comprender las diferentes tecnologías de interconexión existentes sería atender a los

escenarios bajo los cuales son concebidos los sistemas de instrumentación. En este caso, podríamos hablar de tres grandes grupos: instrumentación autónoma o de sobremesa (rack-and-stack), tarjetas de adquisición de datos (pc-based data acquisition) e instrumentación modular (cardcage). Cado uno de los cuales utilizará sendas tecnologías de interconexión comunes o exclusivas de cada arquitectura.

Si hacemos mención a los dispositivos de sobremesa, que abarcan toda la instrumentación electrónica con autonomía de operación, interfaz gráfica propia y que no requieren de la existencia de un controlador externo, éstos pueden ser conectados con un PC a través de algún bus de comunicación como RS-232 (y sus variantes RS-422 y RS-485), bus paralelo, USB, Ethernet/LAN, Firewire o IEEE 1394 y GPIB, principalmente [32], [2],[12].

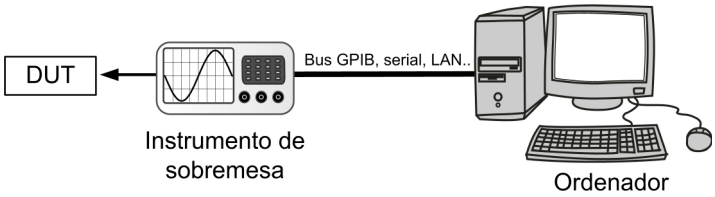


Figura 2.- Esquema clásico de un sistema de sobremesa

Por otra parte, podemos referirnos a las tarjetas de adquisición de datos (TAD), que son aquellos sistemas de instrumentación no autónomos compuestos por circuitos analógicos de entrada para la multiplexación, amplificación, muestreo y retención de las señales de entrada y un conversor analógico-digital, fundamentalmente. Éstos utilizan interfaces de interconexión basadas en ranuras de expansión tipo PCMCIA, PCI, PCI Express y AT, fundamentalmente, aunque recientemente existen TAD con nuevas interfaces como USB y WI-FI, entre otras [33], [12].

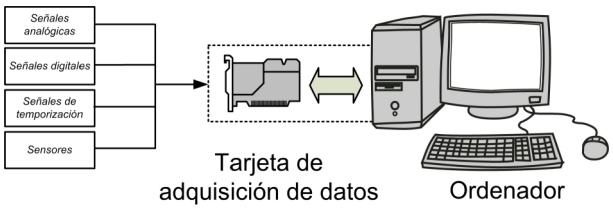


Figura 3.- Esquema clásico basado en tarjetas de adquisición de datos

Otro escenario lo constituyen los dispositivos modulares, es decir, tarjetas de instrumentación que requieren de una infraestructura exterior para funcionar, esto es, un chasis y un controlador. Tal es el caso de la instrumentación VXI, CompactPCI y PXI, entre otras. La instrumentación modular ofrece analizadores de RF, osciloscopios, multímetros, generadores de funciones, etc., del mismo modo que la instrumentación de sobremesa [34]. No obstante, requieren de una infraestructura que permita alojar, administrar y controlar éstos. El control de los instrumentos modulares puede ser realizado empleando un controlador empotrado en la propia estructura modular o bien a través de un PC externo, en cuyo caso se comunica con éste utilizando tecnología GPIB, Firewire, MXI o Ethernet, principalmente [1], [57].

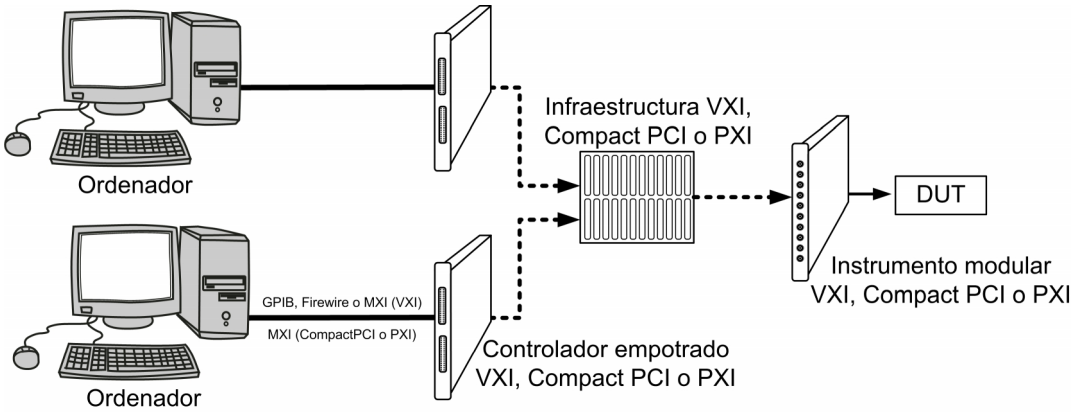


Figura 4.- Sistema de control de instrumentación VXI, CompactPCI y PXI

Por último, otras tecnologías recientemente aparecidas potencian la cohesión de los tres escenarios de instrumentación, tal es el caso de la tecnología LXI, Lan Extensions for Instrumentation, que abre la posibilidad de conexión de estos dispositivos vía LAN [35],[36].

La enorme expansión del número de sistemas de instrumentación aparecidos en los últimos años ha causado una enorme confusión en la comunidad científica en relación a la mejor opción a elegir para implementar una arquitectura de test y medida [37].

En este sentido, algunos autores enfatizan que los aspectos más importantes que se deben tener en cuenta a la hora de elegir tanto una arquitectura como una interfaz de instrumentación son, fundamentalmente, el coste, número de canales, "footprints[2]" velocidad, facilidad de integración y flexibilidad [37].

La instrumentación autónoma o de sobremesa ha sido la única arquitectura existente durante muchos años. Estos instrumentos frecuentemente incorporaban una interfaz de comunicación con el exterior que, en sus comienzos, utilizada el bus serie RS-232 [48], como podemos encontrar en [38], y el bus paralelo, como se contempla en [39], para pronto ser sustituidos por la utilización del interfaz IEEE-488.1 o GPIB, convirtiéndose en un estándar comúnmente aceptado en la industria de la instrumentación electrónica [37], [48], y donde encontramos el mayor abanico de aplicaciones en la actualidad, como podemos comprobar de una simple revisión de la documentación científica, en la que encontramos numerosas aplicaciones al respecto: [21], [40], [41], [42].

La introducción del primer ordenador personal en 1981 por parte de IBM atrajo la atención de muchos proveedores software, convirtiendo el PC en un estándar de facto para aplicaciones tanto personales como industriales. Pese a que inicialmente los PC no fueron considerados una plataforma adecuada para el control de instrumentación debido a las limitaciones de velocidad de los procesadores, en la actualidad han desplazado a otros sistemas de control de instrumentación basados en estaciones de trabajo de compañías como DEC,

[2] hace referencia a la huella o espacio ocupado por la tecnología.

Sun y HP, convirtiéndose en el principal controlador de sistemas de instrumentación [37].

Fue precisamente la aparición del PC de IBM lo que atrajo a numerosas compañías a desarrollar las primeras tarjetas de adquisición de datos que utilizaban las ranuras de expansión de éstos. Estas tarjetas se hicieron muy populares en laboratorios de investigación y educación, surgiendo nuevas aplicaciones que utilizaban este tipo de tarjetas en sistemas de instrumentación, tales como las presentadas en [40], [43] y [44]. No obstante, la mayoría de la instrumentación industrial siguió utilizando la instrumentación autónoma GPIB donde los rangos de frecuencia, precisión y funcionalidad superaban a las TAD [37].

La instrumentación modular aparece a finales de los años 60, cuando HP combina un voltímetro y un multiplexor para crear un escáner. Varios productos de adquisición de datos GPIB aparecieron a lo largo de los años 70 y principios de los 80, incluyendo voltímetros, digitalizadores, multiplexores, switches, sistemas E/S digitales, conversores D/A y generadores de onda. Algunos de éstos incluso contenían tarjetas especiales como controladores de motores paso a paso [37].

En 1985, HP, Tektronix, Wavetek, Ramal-Dana y Colorado Data Systems introdujeron VXI, un estándar de instrumentación modular pensado para aplicaciones militares de estados unidos [2]. Estos sistemas modulares se hicieron muy populares tanto en la industria aeroespacial y militar, con numerosas aplicaciones como las que encontramos en [45] y [46], así como en la industria de fabricación de sistemas de test y adquisición de datos de alta velocidad, como podemos ver en [47], donde la flexibilidad de configuración y el elevado número de canales era necesario [37].

Estos sistemas podían ser controlados externamente con un PC utilizando GPIB o MXI, un bus especial diseñado para la instrumentación VXI. Ambas interfaces, GPIB y MXI, requerían una tarjeta instalada en un PC y otra en el módulo VXI para posibilitar dicha comunicación [37], [48].

Pese a que la instrumentación modular VXI es más costosa, es su flexibilidad y alta tasa de transferencia la que la ha hecho muy popular, alcanzando el 10 % del mercado de sistemas de medida y test [37].

En 1997, National Instrument introdujo un segundo estándar de instrumentación modular, PXI (PCI extensions for Instruments), basado en el bus PCI (y posteriormente PCI express) de los PC [2]. Muchos de los desarrolladores vieron la oportunidad de solucionar los problemas de ruido que suponía utilizar tarjetas dentro del PC, moviendo sus implementaciones, del mismo modo que VXI, a un chasis externo, al mismo tiempo que las implementaciones PXI reducían el espacio ocupado por VXI por cuatro. Todo ello ha contribuido a que PXI abarque un 1.3% del mercado actual, contando con sendas implementaciones de sistemas de instrumentación virtual como podemos encontrar en [49] y [50], por ejemplo.

Por otra parte, la evolución de las tecnologías del PC nos llevan a que, hoy día, éstos poseen otras interfaces como Ethernet, USB y Firewire. Protocolos como DHCP y Universal Plug and Play hacen muy sencillo el

proceso de conexión de un periférico a un PC, lo que ha provocado que, en los últimos años, muchos desarrolladores hayan incorporado estas interfaces en sus sistemas de instrumentación [37]. Sin embargo, estas interfaces están pensadas para su uso en equipos de consumo, careciendo de la robustez e inmunidad al ruido necesarias en multitud de aplicaciones industriales. USB carece de un conector resistente a golpes o vibraciones, por ejemplo, igual que sucede con las otras tecnologías. Pese a ello, estas tecnologías están cada vez más presentes en los sistemas de instrumentación moderna [56], especialmente en sistemas de adquisición de datos (DAQ o DAS), como nos encontramos en [40], [51] y [52], por ejemplo, o elementos complementarios como Webcams USB utilizadas para monitorizar la experimentación [53].

La instrumentación LXI (LAN extensions for Instruments) ha reforzado la idea de conectar la instrumentación a redes de comunicación mediante una estructura modular basada en tecnología LAN. La instrumentación LXI tiene la ventaja de ser de reducido tamaño, modular y autónoma, no precisando de un habitáculo como la instrumentación VXI o PXI para su funcionamiento [54], [55]. En esencia, estos módulos LXI forman una arquitectura de instrumentación modular comunicados con LAN, eliminando la necesidad de costosos habitáculos y uso interfaces como MXI, posibilitando el uso de, únicamente, un PC y una conexión LAN [37], [57].

El interfaz de instrumentación GPIB ofrece, no obstante, la mayor selección de productos del mercado, con más de 10000 productos que abarcan desde voltímetros de bajo coste hasta analizadores de red de altas prestaciones o sistemas de prueba de telefonía celular [37], [48]. De hecho, una de sus mayores ventajas es precisamente que está soportado por la mayoría de los fabricantes de instrumentación del mercado. Empresas como Keithley Instruments, Agilent, Rohde & Schwarz y Tektronix emplean la tecnología GPIB como principal bus de instrumentación en sus aplicaciones [48].

Comparando con VXI y PXI, éstos ofrecen cientos de productos que a menudo difieren de los ofrecidos con instrumentación GPIB. Tienden a digitalizadores multicanal, conversores D/A, conmutadores, entradas/salidas digitales y miles de tarjetas especializadas como Mil Std1553, ARINC 429, CAN bus, aplicaciones de visión, control de motores paso a paso, etc. [37].

Otra de las ventajas de la instrumentación autónoma GPIB es que tienden a ser la instrumentación de mejor precisión debido a que su carcasa está especialmente diseñada y controlada para trabajar en ambientes ruidosos. En este sentido, las TAD son la que peores prestaciones muestran y la instrumentación PXI y VXI ocupa un nivel intermedio. Por otra parte, la instrumentación LAN y LXI proveen de la misma precisión que la instrumentación GPIB [58].

En relación al coste, las TAD constituyen la mejor elección en este sentido, lo que explica su amplia extensión en el ámbito académico e investigador, no obstante, el uso de las TAD implica, obligatoriamente, el uso de un PC. En el caso de la instrumentación modular, PXI es más económica

que VXI, generalmente. En un punto intermedio se encuentra la instrumentación GPIB, convirtiéndose en la mejor solución en relación a la unidad de mérito precio versus prestaciones para instrumentación básica así como la mejor precisión y resolución para productos de mayores prestaciones [37]. La tecnología GPIB es generalmente más económica que otras tecnologías como VXI o PXI frente al mismo nivel de funcionalidad [48].

La instrumentación LAN/USB/Firewire tiene la misma relación coste/prestaciones que la instrumentación GPIB, con la ventaja añadida de no precisar de una tarjeta adicional para posibilitar la comunicación. Como contrapartida, presentan mayores problemas de robustez e inmunidad al ruido [37], [56].

En el futuro, los módulos LXI proveerán una solución de menor coste y amplias prestaciones que, acompañadas de un software adicional conllevará a un nivel de flexibilidad no alcanzado con la instrumentación estándar [37], [54], [55].

Otro aspecto a tener en consideración es la facilidad de integración, difícil de cuantizar debido al elevado número de dimensiones: conexión física, conectividad E/S, aspectos de programación, etc. En relación a la interconexión física, la instrumentación GPIB y LAN son sencillas de instalar físicamente. La instrumentación LAN y LXI eliminan la necesidad de añadir una tarjeta de comunicación en el PC. Por otra parte, tendiendo en cuenta la conectividad E/S, las TAD son las más versátiles, seguidas de la instrumentación VXI y PXI. Cuando se dispone de un número elevado de instrumentos, la instrumentación GPIB requiere un alto número de cables. Igual sucede con la instrumentación LAN y LXI, pero sin la necesidad de instalar nada adicionalmente en el PC. Los cables LAN, USB y Firewire son más maleables que MXI y GPIB, pero menos aptos para entornos ruidosos [56]. Finalmente, desde el punto de vista de la programación, la utilización de drivers se hace fundamental, ya que simplifica el proceso. En este sentido, destaca la instrumentación compatible SCPI, que facilita enormemente la tarea de programación debido a la independencia de los comandos respecto al fabricante de los aparatos. La instrumentación GPIB frecuentemente dispone de drivers y comandos SCPI accesibles, mientras que otras interfaces recaen la labor de programación sobre los propios drivers [37].

En resumen, podemos afirmar que la instrumentación autónoma rack and stack GPIB ha sido la arquitectura predominante desde la aparición de la instrumentación virtual. Nuevas interfaces como LAN, USB y Firewire mejoran la integración con los PCs, pero adolecen de la robustez industrial de otras interfaces [56].

La instrumentación autónoma o de sobremesa, rack and stack, sigue siendo la arquitectura dominante, mientras que la instrumentación modular cubre pequeños módulos de funcionalidad mucho más específica que no están disponibles en la instrumentación autónoma, resolviendo necesidades y aplicaciones de industrias aeroespaciales, de automoción e industriales, principalmente [45], [46], [47], [49], [50].

Pese a que la mayor parte de la instrumentación modular abarca la instrumentación VXI y PXI, algunos sistemas como módulos RF y fuentes de

alimentación especiales, superan las posibilidades de éstas, lo que ha llevado a la aparición de la instrumentación LXI, que es autónoma y modular. La instrumentación LXI abre nuevas posibilidades en el campo de la instrumentación sintética, aumentando la granularidad respecto a la instrumentación autónoma y modular [37].

Los aspectos anteriormente mencionados ponen de manifiesto las virtudes de la instrumentación GPIB frente a otras tecnologías. Su amplia extensión, su facilidad de uso, la disponibilidad de drivers compatibles SCPI, etc., son algunos de los factores que más han influido en su popularidad. Es por ello que, el presente trabajo de investigación, se basa en dichos sistemas de instrumentación.

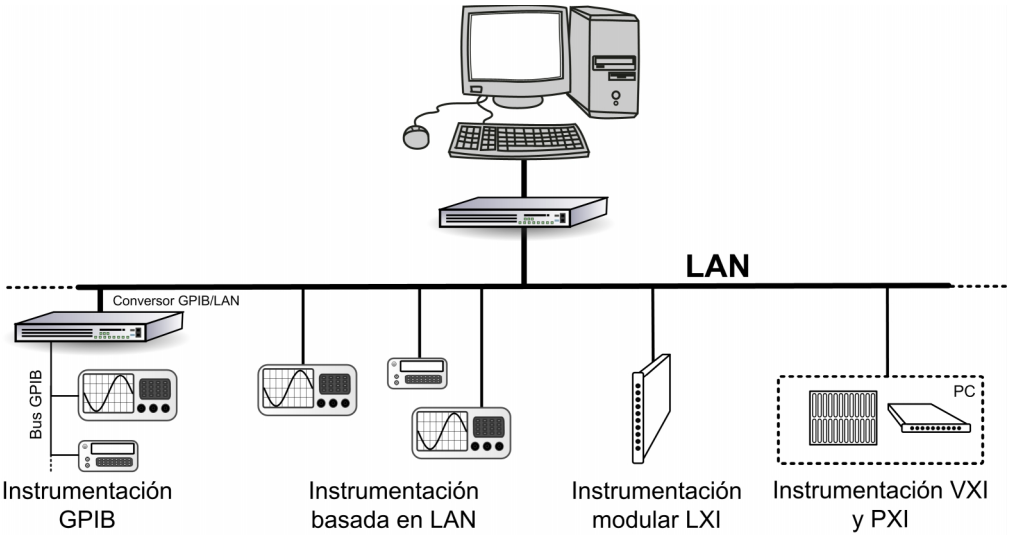


Figura 5.- Esquema de sistemas de instrumentación controlados remotamente

2.4. Software para el diseño y control de instrumentación virtual

Sea cual sea el tipo e interfaz de instrumentación a utilizar, el desarrollo de aplicaciones de instrumentación virtual debe recaer, finalmente, en un determinado software de diseño y control.

Resulta manifiesto que, la eficiencia de cualquier sistema de instrumentación dependerá, no solo del hardware del PC, sino que gran parte de ésta recaerá sobre el software. El software juega un papel fundamental al proveer de una interfaz que actúa como intermediario entre el usuario y los componentes físicos del sistema. Además, el grado de control, flexibilidad, y facilidad de uso dependerán fuertemente de esta interfaz software [59].

Son muchos los factores que deben tenerse en cuenta a la hora de seleccionar un determinado entorno de programación para implementar un sistema de instrumentación virtual. Entre ellos, cabe destacar los siguientes [59].

- Facilidad de uso. Nivel de experiencia requerido para usar el entorno.
- Flexibilidad. Capacidad del paquete software para adaptarse a diferentes requisitos y posibilidad de ser comunicado con otros paquetes software.
- Rendimiento. Criterios y especificaciones de rendimiento del sistema.
- Funcionalidad. Número de funciones y correcta interfaz con la configuración del hardware elegido.

Facilidad de uso

Si bien es cierto que la figura de mérito “facilidad de uso” dependerá, en gran medida, de la persona (o personas) que utilicen el sistema, existen algunas premisas que, de forma generalizada mejoran la calidad de uso del sistema. Una de ellas hace mención a la necesidad de crear entornos intuitivos, procurando que el usuario se encuentre con situaciones inesperadas. De igual forma, puede afirmarse que resulta importante imitar fielmente los controles e indicadores con los que el usuario está familiarizado, contribuyendo a la mejora de la facilidad de uso del entorno [59]. En este sentido, existe una tendencia generalizada hacia la utilización de entornos gráficos, lo que ha contribuido exponencialmente a la mejora de los entornos de instrumentación gracias a la incorporación de interfaces de usuario WIMP (Windows, icons, and pull-down menus), es decir, basados entornos de ventanas con iconos y menús [59].

Flexibilidad

Algunos paquetes software, particularmente aquellos escritos para aplicaciones específicas, tienden a adolecer de una flexibilidad suficiente a la hora de diseñar aplicaciones, dado que se basan en modelos predeterminados. Esto supone limitaciones en las opciones que podemos proporcionar al operador del entorno y en la necesidad, en muchos casos, de modificar la aplicación a bajo nivel, incluso a nivel de código fuente [59].

Otros paquetes software más flexibles permiten al usuario configurar el software para un hardware particular o, alternativamente, disponen de la opción plug and play. En todos los casos, es importante facilitar el proceso de instalación y configuración, haciéndolo transparente al usuario [59].

Un aspecto fundamental desde el punto de vista del usuario es la posibilidad de guardar e importar/exportar datos de un entorno a otro, lo que posibilita su procesado con aplicaciones estadísticas y el análisis gráfico. La capacidad de intercambio de datos DDE (Dynamic Data Exchange) ha hecho posible intercambiar datos de forma transparente entre aplicaciones de tal forma que, por ejemplo, es posible que una aplicación a medida genere datos en formato Excel sin necesidad de exportarlos o importarlos previamente [59].

Rendimiento

El rendimiento de un paquete software es, a menudo, difícil de estimar, dado que los factores determinantes del mismo pueden variar de una aplicación a otra. La velocidad de procesamiento, por ejemplo, será importante en algunas aplicaciones, caso de un osciloscopio virtual, pero irrelevante en otras [59].

Funcionalidad

La funcionalidad se está convirtiendo en un factor de mérito cada vez más importante a la hora de elegir un paquete software. Y se refiere a lo apropiado y aptitud de un programa para una determinada aplicación y configuración software. La mayoría de los paquetes software ofrecen funcionalidades adicionales que no forman parte de la instalación o paquete básico [59], lo que implica diferencias sustanciales en el coste de la aplicación final.

Si bien algunos autores clasifican los paquetes software atendiendo a aspectos relativos meramente a la facilidad de uso y flexibilidad [1], o simplemente hacen mención a la característica de programación textual o gráfica del software [60], podemos afirmar que, la mayoría de los paquetes software aplicables al diseño, implementación y control de sistemas de instrumentación virtuales pueden ser clasificados atendiendo a las anteriores figuras de mérito [59], tal y como se muestra en la siguiente tabla.

Facilidad de uso	Flexibilidad	Rendimiento	Funcionalidad	Ejemplos
Muy complejos (requieren un elevado nivel de conocimientos de programación)	Diseñados para satisfacer un conjunto particular de requisitos. Altamente customizable.	Puede ser muy rápido y compacto pero requiere de un tiempo de desarrollo elevado.	Las requeridas para satisfacer las dadas por una especificación establecida.	Lenguajes de programación genéricos tales como MASM32, Turbo C++, PowerBasic, Visual Basic, Visual C++, etc.
Moderadamente complejos (requiere un conocimiento moderado de programación	Flexible y adaptable. Ofrecen customización significativa	Rápidos y eficientes. Requieren un tiempo de desarrollo significativo	Ofrecen un elevado nivel de funcionalidad	Software escrito a medida usando lenguajes de programación en conjunción con extensiones, librerías externas, controles Active X, etc.
Nivel medio de complejidad	Moderadamente flexible y adaptable a la mayoría de las situaciones. Moderadamente customizable	Raoznablemente rápidos y moderadamente eficientes. Desarrollo de la aplicación puede ser razonablemente rápido	Ofrece un elevado grado de funcionalidad pero puede requerir configuración para una determinada aplicación	Aplicaciones programables y herramientas de análisis de datos, tales como **LabVIEW,** Dasylab, etc.
Simple y relativamente	Limitados y bastante	Usualmente rápidos y	Los niveles de funcionalidad	Aplicaciones de uso

fácil de usar. No requiere experiencia de programación	inflexibles. Customización limitada	eficientes. No se requiere tiempo de desarrollo	varían pero a menudo están limitados a una función en particular	específico, como TracerDAQ, WinDAQ, etc.

Tabla 1.- Clasificación de los paquetes software empleados en instrumentación

Atendiendo a lo anterior, un primer grupo lo constituye el conjunto de entornos para el control de un instrumento específico o tarjetas de adquisición de datos, es decir, las aplicaciones de uso específico [1], [59]. Son entornos diseñados para resolver un problema o conjunto de problemas particular [59]. Permiten, mediante una interfaz de menús desplegables, configurar y programar el dispositivo para la adquisición de una señal y su visualización, pero carecen de la posibilidad de programar con algún lenguaje. Algunos ejemplos de estos entornos son el software DAQ-ware que National Instruments ofrecía con sus tarjetas de adquisición datos, actualmente sustituido por un driver denominado NI-DAQmx, accesible por varios entornos de programación

[61], y con los cuales podemos encontrar algunas aplicaciones en la bibliografía científica [62], [63], [64]. Otro paquete software es TracerDAQ, de la firma Measurement Computing y con un coste aproximado de 200 dólares [65], del cual también podemos encontrar algunas aplicaciones dentro del sector de la instrumentación [66], [67]. Por último, podemos mencionar el paquete software WINDAQ, de DATAQ Instruments [68], con versiones limitadas gratuitas y con un coste de 2000 dólares para la versión más completa, con posibilidad de añadir complementos a los mismos, y empleada en diversas aplicaciones como podemos ver en los trabajos referenciados en [69], [70], entre otras muchas aplicaciones [1] , [59].

El segundo bloque lo constituyen aquellos paquetes software que se basan en lenguajes de programación genéricos o específicos, y que se denominan comúnmente entornos de programación lingüísticos o textuales [1], [60].

Dentro del segundo bloque, a su vez, podemos diferenciar aquellos entornos de programación lingüísticos basados en entornos gráficos, caso de LabWindows™/CVI, por ejemplo, o bien los meramente textuales, como Turbo C. No obstante, la programación íntegra de entornos basados en línea de comandos ha quedado en desuso en la actualidad.

Los entornos de programación lingüísticos basan su principio de funcionamiento en un determinado lenguaje de programación, este puede ser estándar (C, BASIC, etc.) o propio del entorno; procedimental u orientado a objetos [60]. Los entornos de programación textuales pueden ser entornos genéricos utilizados para múltiples aplicaciones, lo que implica que, para acceder al hardware de instrumentación es preciso realizarlo a muy bajo nivel, accediendo a los registros del sistema o bien a nivel de driver [12]. Existe, no obstante, una tercera posibilidad, basada en el uso de lo que se denominan extensiones del lenguaje. Las extensiones del lenguaje son componentes que se pueden adicionar al entorno, frecuentemente propietarios y costosos. Estas extensiones incluyen librerías dinámicas, bloques compilados, etc. Una tendencia generalizada en la actualidad es la utilización de controles Active X.

Estos controles, definidos por Microsoft, describen componentes software modulares y reutilizables que pueden ser utilizados por cualquier entorno que soporte este estándar. Por ejemplo, podríamos utilizar controles Active X de la firma DATAQ Instruments, sin modificación, por Visual Basic, Visual C++, Borland C++, etc., para simplificar la tarea de programación de una consola de monitorización de datos frente al tiempo utilizando dichos componentes [59].

Por otra parte, los entornos de programación lingüísticos también pueden ser específicos u orientados a instrumentación virtual, caso de LabWindows™/CVI, por ejemplo [1], [18]. En este caso se dispone de librerías de funciones para el análisis y la presentación de datos en el propio entorno. Incorporando, en muchos casos, una interfaz gráfica de menús desplegables que permite el acceso a estas funciones para facilitar la generación del programa [1].

Un aspecto importante de los entornos de programación lingüísticos a tener en cuenta es el modo de ejecución de la aplicación final, dado que algunos entornos permiten únicamente la ejecución de la aplicación dentro del mismo. Esto tiene como ventaja las facilidades de depuración que incluyen y la simplificación de la interfaz con el hardware del controlador. En contrapartida, la velocidad de ejecución es mucho menor y requiere el pago, en la mayoría de los casos, de una licencia run-time para su distribución o venta [1].

Han sido muchos los entornos de programación lingüísticos existentes a lo largo de la historia de la instrumentación virtual que han quedado en desuso, caso de VisuaLab, de la firma IOtech, que era una extensión de Visual Basic, Asyst, de la empresa Keithley, que utilizaba un lenguaje propio y funcionaba bajo entornos MS-DOS, HPITG II, de Hewlett Packard, etc [1]. En la actualidad, perduran otros tantos entornos de programación lingüísticos, entre los que podemos destacar los siguientes [1], [60]:

LabWindows™/CVI (National Instruments): Teniendo su origen en 1986, se ejecutaría bajo entorno MS-DOS inicialmente, y fue denominado simplemente Labwindows. La inclusión de las siglas CVI (C for Virtual Instrumentation) fueron añadidas en la primera versión compatible con entornos Windows y Solaris. LabWindows™/CVI es un entorno de desarrollo integrado basado en ANSI C, es decir, procedimental, que proporciona un extenso juego de herramientas y librerías orientadas a la instrumentación virtual [18]. Las últimas versiones ofrecen compatibilidad con objetos Active X, interfaces de instrumentación y puertos de comunicación, así como innumerables módulos adicionales que permiten la interfaz con múltiples dispositivos e interfaces [1] ,

[71]. Su coste aproximado, abarca desde los 1250 Euros, para la versión más básica en entorno Windows y los 3100 Euros para versiones intermedias, a lo que habría que añadir el coste de los módulos adicionales [72]. En la actualidad, son muchos los autores que han desarrollado aplicaciones de instrumentación empleando el entorno LabWindows™/CVI, entre los que podemos encontrar los referenciados en [73], [74] y [75], entre otros.

Matlab/Simulink (MathWorks). Matlab es un lenguaje de programación procedimental de propósito general que, con la inclusión de los módulos

adicionales “Data Acquisition Toolbox” e “Instrument Control Toolbox” [76] ha sumado su potencia de cálculo y herramientas del entorno a las capacidades de diseño de sistemas de instrumentación virtual, soportando interfaces VISA, GPIB, TCP/IP e incluso UDP. No obstante, la flexibilidad, portabilidad e intefaz gráfica de Matlab son muy limitadas cuando son comparadas con entornos como LabWindows™/CVI y LabVIEW, entre otros [60]. Una revisión a la página Web de la firma nos puede orientar hacia el coste de la herramienta, cuya licencia profesional ronda los 2000 Euros, a lo que habría que sumar 1000 Euros por cada uno de los Toolbox mencionados anteriormente [76]. También podemos encontrar implementaciones de sistemas de instrumentación virtual basados en Matlab, como son [77], [78] y [79].

Los entornos de programación visuales a menudo se asocian con la programación orientada a objetos. Si bien es cierto que algunas aplicaciones de programación visual proporcionan al usuario funcionalidad orientada a objetos, la programación visual no está intrínsicamente orientada a objetos, la utilización de un control o gráfica en una pantalla no es criterio suficiente ni necesario para considerar una aplicación orientada a objetos. Además, las aplicaciones con entornos tipo WIMP pueden no estar realizadas con lenguajes orientados a objetos, de hecho, la mayoría de los productos existentes fueron construidos con herramientas procedimentales tradicionales [81].

Tanto Labwindows/CVI como Matlab son entornos de programación procedimental. Si bien es cierto que el paradigma de programación procedimental es el más común y extendido en general, este hecho no se cumple cuando observamos el dominio de la instrumentación virtual [60].

En los lenguajes procedimentales un programa se diseña en base a un conjunto de instrucciones o sentencias, de tal manera que cada sentencia o instrucción indica al compilador la realización de alguna tarea: obtener una entrada, producir una salida, etc. [60]. En programas pequeños con escasa complejidad este paradigma se manifestaba eficiente. Un programador únicamente ha de crear un conjunto de instrucciones en dicho lenguaje de programación para posteriormente compilarlos y ser ejecutador en el ordenador [80]. La necesidad de aumentar la extensión y complejidad de los programas llevó a la descomposición de éstos en unidades más pequeñas, denominadas funciones, procedimientos, subprogramas o subrutinas. De esta manera, los programas se diseñaban mediante un conjunto de funciones, cada una de las cuales tiene un propósito bien definido y resuelve una tarea concreta, diseñando una interfaz claramente definida para su comunicación con otras funciones. Un paso adelante en esta misma dirección se alcanzó con el agrupamiento de funciones en unidades superiores denominadas módulos (en el caso de C++ se concretaban en archivos o ficheros); no obstante, el paradigma de programación continuaba siendo el mismo: agrupar componentes que ejecutan listas de instrucciones (sentencias) [60], [80].

Pese a su evolución y ventajas, como su relativa simplicidad y facilidad de implementación para compiladores e intérpretes, los programas estructurados presentan una serie de debilidades: de un lado, las funciones tienen acceso ilimitado a los datos globales; por otro, las funciones inconexas y datos, fundamentos del paradigma procedimental, proporcionan un modelo “pobre” del mundo real, especialmente en el campo de la instrumentación

virtual, lo que la hace menos eficiente frente a otros paradigmas de programación [60], [80].

Fruto de las limitaciones manifiestas de la programación procedimental surge la necesidad de incorporar la programación orientada a objetos en los nuevos enfoques de programación de sistemas de instrumentación virtual [60].

La programación orientada a objetos (POO u OOP) aporta un nuevo enfoque. Al contrario que la programación procedimental, cuyo acento se encuentra en los algoritmos, la POO enfatiza en los datos, ajustando el lenguaje al problema. De esta forma, el paradigma orientado a objetos diseña formatos de datos que se corresponden con las características esenciales de un problema. Los lenguajes orientados combinan en una única unidad o módulo, tanto los datos como las funciones que operan sobre esos datos. Tal unidad se llama objeto. Si se desea modificar los datos de un objeto, hay que realizarlo mediante las funciones miembro del objeto. Ninguna otra función puede acceder a los datos. A su vez, la comunicación entre objetos se produce mediante mensajes. Esto simplifica la escritura, depuración y mantenimiento del programa [60], [80].

Los lenguajes OOP empleados en instrumentación virtual incluyen Visual C++, Visual C#, Java y Visual Basic.NET, principalmente. Visual Basic.NET forma parte de una plataforma más amplia basada en el .NET Framework, y es orientado a objetos. Esto marca una diferencia sustancial con las versiones anteriores de Visual Basic [60], [81], [82]. En la bibliografía encontramos algunos ejemplos de sistemas de instrumentación virtual que basan su diseño en dichos lenguajes de programación [83], [84] y [85].

En relación a Java, pese a ser un lenguaje orientado a objetos, su uso en el acceso al hardware de sistemas de instrumentación virtual ha sido relegado a aplicaciones RS-232 e instrumentación LXI, principalmente. De hecho, y para extender las capacidades de los sitios Web para interactuar con instrumentación, recientemente ha aparecido una extensión de Java para tareas de comunicación denominada Java Communication API. Esta API posibilita el acceso a los dispositivos con interfaz serie, paralelo, USB y LXI [60].

En el caso de LXI, Agilent y Keithley han desarrollado API de comunicación Java propietarias que proveen prestaciones Web a los instrumentos, permitiendo a los usuarios operar con los instrumentos vía Web de forma similar a cómo usan un panel frontal. Con un servidor Web incorporado dentro de sus equipos LXI y un navegador Web con soporte Java, el instrumento puede ser completamente controlado mediante una red LAN usando una interfaz Web [60].

Tecnologías Web como los Applets Java pueden proporcionar interfaces Web sencillas de usar, con pocos requisitos de memoria y procesado [60].

Hoy día, el hardware de instrumentación viene acompañado de drivers software para Visual C++, Visual C#, Visual Basic y Visual Basic.NET principalmente. Keithley y National Instruments ofrecen diferentes paquetes software compatibles con otros lenguajes. Tal es el caso de Measurement Studio de National Instruments, un juego integrado de clases y controles para

aplicaciones de test, medida y automatización en Microsoft Visual Studio. Measurement Studio, con un coste que abarca desde los 500 Euros en su edición más limitada hasta los 2600 euros de su edición empresarial, reduce el tiempo de desarrollo de aplicaciones proporcionando componentes de interfaz usuario como son formularios Windows, formularios Web y Active X [60], [86].

Por último, el tercer grupo lo constituyen los entornos de programación gráficos. Siendo su aparición en el mercado la más reciente, el paradigma de diseño de aplicaciones es totalmente diferente, permitiendo crear aplicaciones con un cierto grado de complejidad mediante la unión de iconos de forma totalmente gráfica y siguiendo una estructura jerárquica. Estos iconos son bloques de módulos software preprogramados tanto genéricos como específicos que tienen accesibles sus propiedades para poder utilizarlos, disponiendo de iconos que nos permiten crear interfaces WIMP emulando los controles de instrumentación convencional. Análogamente a los lenguajes de programación clásicos, se dispone de múltiples tipos de datos y estructuras de programación como bucles, condiciones, E/S, etc. Además, algunos entornos incorporan un compilador gráfico para aumentar la velocidad de ejecución [1], [60].

El ciclo de aprendizaje y programación se reduce drásticamente al ser entornos que imitan una forma de programación muy parecida a un diagrama de flujo o bloques. Esta simplificación en la forma de programación lleva asociada algunas limitaciones. La velocidad final de la aplicación será mucho menor que su versión en lenguaje C y la utilización de muchos elementos gráficos e iconos requiere grandes cantidades de memoria y potencia de cálculo [1], [60].

Uno de los primeros y más conocidos entornos de programación gráfica es LabVIEW: Laboratory Virtual Instrument Engineering Workebench, de la firma National Instruments. Su primera versión data de 1986 [1]. LabVIEW permite el diseño de interfaces de usuario mediante una consola interactiva en la que el diseñador dispone de una completa gama de librerías de iconos para la manipulación de datos, control de flujo, interfaz de usuario (botones, gráficos, menús, etc.), presentación y almacenamiento de datos, tarjetas de adquisición de datos del propio fabricante y drivers de la mayoría de instrumentos controlables (GPIB, VXI, RS-232, CAMAC) disponibles en el mercado [2], [1].

Presentando un entorno similar al que nos encontramos en entornos lingüisticos, LabVIEW se basa en programación gráfica o lenguaje G para crear programas basados en diagramas de bloques. También pueden incluirse rutinas en lenguaje C como iconos y llamadas a funciones de librerías externas, por ejemplo DLL de MS-Windows. Existe un elevado número de fabricantes de hardware y software que desarrollan y mantienen centenares de bibliotecas para LabVIEW y drivers de instrumentos que le ayudan a utilizar fácilmente sus productos con este. No obstante, ésta no es la única forma de proporcionar conectividad entre aplicaciones, también ofrece mecanismos para incorporar programas en ActiveX, bibliotecas dinámicas (DLLs) y bibliotecas compartidas de otras herramientas [10], [15], [2].

Cuenta, en la actualidad, con versiones para entornos Windows, Linux y MAC Os y Sun Solares que abarcan un coste desde los 1250 Euros para la

versión más básica hasta los 4300 Euros para la más completa [99], a lo que habría que añadir el coste de los módulos adicionales. Los instrumentos virtuales que se crean en una plataforma pueden ser transportados de manera transparente a cualquier otra plataforma simplemente abriendo y ejecutando el programa, no obstante, la aplicación final corre dentro del entorno, siendo interpretado en tiempo de ejecución y es necesario el pago de una licencia run-time para aplicaciones comerciales [1], [60].

LabVIEW constituye la plataforma más extendida para el desarrollo de sistemas de instrumentación virtual, pudiendo encontrar infinidad de aplicaciones presentes en la comunidad científica tales como [87], [88], [89], [90] y [91], entre otras muchas.

Otro exponente de los entornos de programación gráfica actuales es Agilent VEE, cuya apariencia y modo de funcionamiento sigue fielmente al entorno LabVIEW. VEE (Visual Engineering Environment) dispone de varias versiones con un coste en el rango de los 640 euros a 1440 euros [100]. En la actualidad soporta plataformas Windows. Se trata de un entorno gráfico basado en la programación mediante diagramas de flujo, a igual que LabVIEW, lo que lo hace fácil de manejar e intuitivo. Se ejecuta dentro del propio entorno y es interpretado en tiempo de ejecución, del mismo modo que LabVIEW. No obstante, la principal diferencia entre VEE y LabVIEW se encuentra en la arquitectura de la programación gráfica precisamente, ya que, mientras que VEE personaliza sus iconos con una etiqueta, LabVIEW no los modifica necesitando recordar el significado de cada uno de ellos. Permite el control de interfaces GPIB, LAN, USB, RS-232, VXI, PXI, etc. VEE puede manejar diferentes tipos de datos, siendo posible utilizar las funciones de VEE por otros programas a través de ActiveX, soportando varios lenguajes de programación como son Visual Basic, C/C++, Matlab, Visual C#, los lenguajes .NET, e incluso LabVIEW, y es compatible con otros programas Microsoft, tales como Word, Excel, Outlook, etc. [60].

Algunos desarrollos de instrumentación virtual presentados a la comunidad científica son los que encontramos en [92], [93] y [94].

Además de LabVIEW y Agilent VEE podemos encontrar en el mercado otras herramientas gráficas con un paradigma de funcionamiento similar, siguiendo una estructura gráfica basada en diagramas de flujo y jerárquicas, tal es el caso de, por ejemplo, Measure Foundry, de la firma Data Translation [95], Testpoint, software de la empresa Measurement Computing [96], Dasylab, de National Instruments [97], entre otras [60].

2.5. Arquitecturas de acceso remoto en sistemas de instrumentación

La implementación de un sistema de instrumentación virtual tiene como objetivo el acceso a un determinada aparamenta mediante el empleo de alguna de las interfaces de instrumentación y entornos software anteriormente descritos. Si bien es cierto que la interfaz de usuario puede estar disponible desde el ordenador local donde se ejecuta la aplicación que actúa de interfaz con los aparatos de instrumentación, un paso adicional sería el acceso remoto a dicho sistema a través de una determinada red de comunicación desde un punto distinto al computador local que soporta la interfaz de instrumentación

[60]. Esto ha dado lugar a lo que se denomina instrumentación virtual remota, término frecuentemente abreviado en la bibliografía científica y técnica como instrumentación remota o simplemente, instrumentación virtual, renombrando a la anterior definición [101].

Desde un punto de vista de la ingeniería del software, los sistemas de instrumentación remota constituidos por varios computadores que comparten información sobre una red de comunicación son una particularización de lo que se denomina "sistema distribuido". Un sistema distribuido es un sistema en el que el procesamiento de la información se distribuye sobre varias computadoras en lugar de estar confinado a una única máquina [102].

El diseño arquitectónico de un sistema distribuido de instrumentación debe afrontar, en primer lugar, el tipo de arquitectura de sistema distribuido a diseñar. Una primera distinción suele hacer referencia al marco bajo el cual tiene lugar el intercambio de información, distinguiéndose entre arquitecturas intraorganizacionales e interorganizacionales. En el primer caso, la distribución de las aplicaciones e información se lleva a cabo dentro de una sola organización. Las dos arquitecturas intraorganizacionales más significativas son la arquitectura cliente-servidor y la arquitectura de objetos distribuidos. La arquitectura cliente-servidor presenta un modelo con un conjunto de servicios que se proporcionan a los clientes que utilizan dichos servicios, habiendo dos estructuras claramente diferenciadas: Clientes y servidores. Por otra parte, la arquitectura de objetos distribuidos no presenta distinción entre clientes y servidores, siendo un modelo basado en un conjunto de elementos que interaccionan y cuya localización es irrelevante, no habiendo distinción entre proveedores de servicios y usuarios de los mismos. Por otra parte, Las arquitecturas interorganizacionales, más adecuadas para sistemas distribuidos entre organizaciones, se dividen en dos, la arquitectura peer-to-peer y arquitectura orientada a servicios [102].

2.5.1. Arquitecturas cliente-servidor

El paradigma cliente-servidor responde a un modelo que se organiza como un conjunto de servicios y servidores asociados, junto con unos clientes que acceden a dichos servicios [102]. Los principales componentes de este modelo son:

- Un conjunto de servidores ofreciendo servicios a otros subsistemas.
- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.
- Una red que permite a los clientes acceder a estos servicios.

En este modelo, los clientes precisan conocer los nombres de los servidores disponibles y los servicios que éstos proporcionan. En contraposición, los servidores no precisan conocer la existencia o identidad de los clientes, que acceden a los servicios proporcionados por un servidor a través de llamadas a procedimientos remotos usando un protocolo de petición-respuesta tal como el protocolo http usando en la WWW. Básicamente, un cliente realiza una petición a un servidor y espera hasta que recibe una respuesta. Varios procesos servidores pueden ejecutarse sobre un único

procesador; por lo tanto, no hay necesariamente correspondencia 1:1 entre procesos y procesadores en el sistema [102].

La estructura lógica de cualquier aplicación cliente-servidor puede dividirse en 3 capas: Presentación, Procesamiento de la aplicación y Gestión de datos. La primera de ellas hace referencia a la presentación de la información al usuario y su interacción con él, es decir, abarca la interfaz de usuario. La capa de procesamiento constituye la lógica propia de la aplicación. Por último, la capa de datos hace referencia a las operaciones precisas con bases de datos [102].

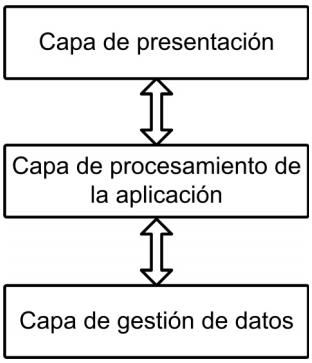


Figura 6.- Torre de capas de las aplicaciones

La arquitectura cliente-servidor más simple es la arquitectura de dos capas, en la que el sistema se organiza como un servidor (o varios idénticos) y un conjunto de clientes. En estas arquitecturas la capa de presentación se encuentra ubicada en el cliente, mientras que la capa de gestión de datos se encuentra en el servidor. En función de la ubicación de la capa de procesamiento: en el cliente o servidor, podemos hablar de los modelos de cliente rico o ligero, respectivamente [102]. En el modelo de cliente ligero (thin client) las capas de procesamiento y gestión de datos tienen lugar en el servidor, siendo el cliente únicamente responsable de la capa de presentación de la aplicación. Por otra parte, el modelo de cliente rico (fat client), libera al servidor de la capa intermedia y éste únicamente se ocupa de la gestión de datos, siendo el software cliente el encargado de la lógica de la aplicación y las interacciones con el usuario del sistema [102].

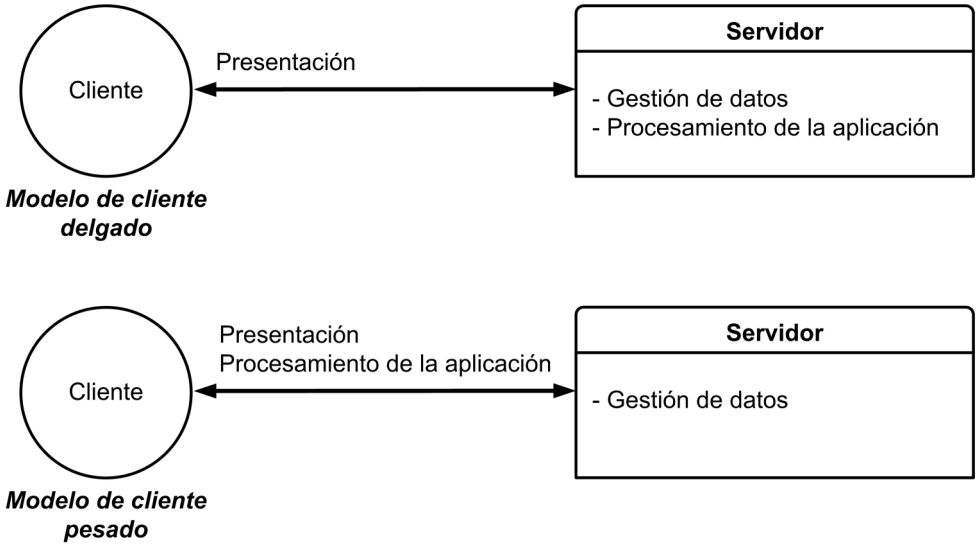


Figura 7.- Arriba: Modelo de cliente ligero. Abajo: Cliente rico o pesado

El modelo de dos capas se utiliza, frecuentemente, cuando un sistema centralizado, funcionando localmente, evoluciona hacia una arquitectura cliente-servidor. Este es el caso más común en los sistemas de instrumentación virtual, que han evolucionado paulatinamente a sistemas remotos, encontrando multitud de aplicaciones de instrumentación virtual que siguen el modelo de 2 capas, tales como [104] y [105], entre otros. La interfaz con el usuario es la que se lleva a un PC remoto, pudiendo implementarse vía un navegador Web, inclusive, mientras que la aplicación en sí misma actúa como servidor, gestionando procesamiento y datos [102].

Encontramos sendos ejemplos de arquitecturas de instrumentación desarrolladas siguiendo este modelo de cliente basado en un simple navegador Web, siendo una de la opciones más empleadas en la actualidad, dada la extensión del uso de navegadores Web hoy día.

El mayor problema de un paradigma de cliente ligero radica en la elevada carga de procesamiento que recae sobre el servidor, responsable de toda la algorítmica y el tráfico generado entre cliente y servidor, dado que el servidor es el responsable de todos los cálculos y esto puede sobrecargar la red [102]. Aún así, la utilización del paradigma de cliente ligero presenta otras tantas ventajas y es utilizado en arquitecturas propuestas en la bibliografía científica, como podemos ver en [106].

El paradigma de cliente rico aprovecha la potencia de procesamiento del cliente y éste asume dos capas del modelo, procesamiento e interfaz de usuario, siendo el servidor un mero sistema de transacción con la base de datos. No obstante, pese a esta distribución de la capacidad de procesamiento, la gestión del sistema se complica, dado que la funcionalidad del sistema queda expandido a varios computadores, y esto presenta serias dificultades cuando el entorno tiene que ser modificado, debiendo reinstalar el software en cada cliente [102], lo que supone un coste elevado, al tiempo que puede presentar serias limitaciones, como son la falta de privilegios de administrador del sistema de los usuarios para proceder a dicha reinstalación o la carencia de conocimientos en la forma de proceder.

En este sentido, la aparición de lo que se denomina “código móvil”, caso de los applets de Java y los controles Active X, que pueden descargarse en un cliente desde un servidor, ha permitido el desarrollo de sistemas cliente-servidor que son una solución de compromiso entre los paradigmas de cliente ligero y rico, de manera que esta tecnología permite descargar en el cliente parte de la capa de procesamiento, en forma de código móvil, soliviantando la carga del servidor [84]. Esto ha dado lugar a numerosas implementaciones que siguen este modelo [52], [105], [107], En este caso, la interfaz de usuario se realiza vía un navegador Web que incluya las utilidades de construcción de programas para ejecutar el código descargado [102].

El mayor inconveniente del paradigma de dos capas es que las tres capas lógicas deben asociarse a dos computadoras: cliente y servidor. Esto puede suponer problemas de escalabilidad, rendimiento y gestión del sistema. Para soslayar estos problemas, una alternativa es utilizar una arquitectura cliente-servidor de tres capas. En esta arquitectura, las tres capas lógicas: Presentación, procesamiento y gestión de datos, son procesos lógicos separados ejecutados en máquinas diferentes, lo que permite aumentar la escalabilidad del sistema, debido a que es relativamente fácil añadir nuevos servidores a medida que el número de clientes crece. Además, para manejar la recuperación de la información del servidor de gestión de datos, que comúnmente será una base de datos, se utilizará un middleware que soporte consultas a la base de datos en algún lenguaje como, por ejemplo, SQL (Structured Query Language) [102].

Podemos encontrar algunas propuestas en la bibliografía científica que intentan aproximarse al modelo de 3 capas anteriormente expuesto en un intento de potenciar la escalabilidad del sistema, así como su rendimiento y gestión. Un ejemplo lo encontramos en [22], donde se propone una arquitectura de tres capas con un cliente que únicamente gestiona la interfaz de usuario, un primer servidor (RLS: Real Laboratory Server) que gestiona las interacciones con la instrumentación electrónica y el procesamiento de la aplicación, y un tercero (VLS: Virtual Laboratory Server) encargado de la gestión de los datos, implementando la política de acceso, usuarios, claves de acceso, etc. Otro ejemplo de un sistema de instrumentación remoto de 3 capas lo encontramos en [106].

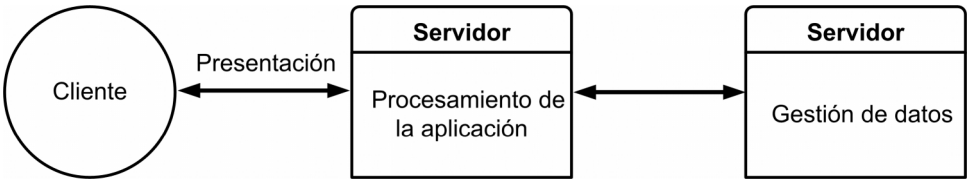


Figura 8.- Arquitectura cliente-servidor de tres capas

También podemos encontrarnos con la extensión del modelo de tres capas a una variante multicapas en la que se añade al sistema servidores adicionales. Esto puede ser útil cuando los usuarios precisen acceder y usar datos de diferentes bases de datos, en cuyo caso un servidor de integración se ubica entre el servidor de aplicaciones y los servidores de gestión de datos. El servidor de integración recogerá los datos y los presentará a la aplicación como si proviniesen desde una única base de datos [102].

El paradigma de tres capas y las variantes multicapas que distribuyen el procesamiento de la aplicación entre varios servidores son intrínsecamente más escalables que los modelos de dos capas. El tráfico de red se reduce, al contrario de lo que sucedía con las arquitecturas de dos capas de cliente ligero. La capa de procesamiento es la parte más volátil del sistema, siendo fácilmente actualizable debido a que está centralizada. El procesamiento también puede distribuirse entre la lógica de la aplicación y los servidores de gestión de datos, en cuyo caso permite una respuesta más rápida a las peticiones de los clientes [102].

Serán los diseñadores los que deberán tener en cuenta una serie de factores cuando eligen la arquitectura más adecuada.

Arquitectura	Aplicaciones
C/S de dos capas con clientes delgados	Aplicaciones de sistemas heredados donde no es práctico separar el procesamiento de las aplicaciones y la administración de datos. Aplicaciones computacionalmente intensivas como los compiladores con poca o ninguna administración de datos. Aplicaciones intensivas en datos (navegar y consultar) con poco o ningún procesamiento de la aplicación.
C/S de dos capas con clientes gruesos	Aplicaciones con procesamiento de datos computacionalmente intensivo (por ejemplo, visualización de datos, animaciones gráficas,...) Aplicaciones con funcionalidad para el usuario final relativamente estable utilizadas en un entorno con administración de sistemas bien establecido.
C/S de tres capas o múltiples capas	Aplicaciones de gran escala con cientos o miles de clientes. Aplicaciones donde tanto los datos como la aplicación son volátiles. Aplicaciones donde se integran datos de diversas fuentes.

Tabla 2.- Utilización de diferentes arquitecturas cliente-servidor

En resumen, podemos decir que la ventaja principal del modelo cliente-servidor radica en que se trata de una arquitectura distribuida, siendo posible disponer de varios procesadores distribuidos, añadir con facilidad un nuevo servidor e integrarlo, así como actualizar un servidor de forma transparente sin afectar al resto del sistema [102]. Además, resulta sencillo conectar diferentes plataformas y sistemas operativos, presenta una capacidad ilimitada y un número de usuarios también ilimitados. El modelo cliente-servidor se organiza como un conjunto de servicios proporcionados por unos servidores y un grupo de clientes usuarios de dichos servicios.

Un repaso a la bibliografía científica pone de manifiesto que, la casi totalidad de las arquitecturas de instrumentación virtual remota atienden al paradigma cliente/servidor [60].

2.5.2. Arquitecturas de objetos distribuidos

Uno de los problemas de las arquitecturas cliente-servidor es que limitan la flexibilidad de los diseñadores, ya que ellos deben decidir dónde ubicar los servicios proporcionados. Unido a lo anterior, también se debe planificar la escalabilidad y proveer de algún método para distribuir la carga sobre los servidores cuando el número de cliente aumenta, tal y como hemos comentado anteriormente [102].

Un acercamiento más general al diseño de sistemas distribuidos sería eliminar la distinción entre cliente y servidor y diseñar la arquitectura del sistema en base a unos objetos distribuidos. En un paradigma de objetos distribuidos, los elementos principales del sistema son objetos que proporcionan una interfaz a un conjunto de servicios que ellos suministran. Otros objetos realizan llamadas a estos servicios sin hacer distinción lógica entre un servidor (el proveedor del servicio) y un cliente (el receptor de un servicio) [102].

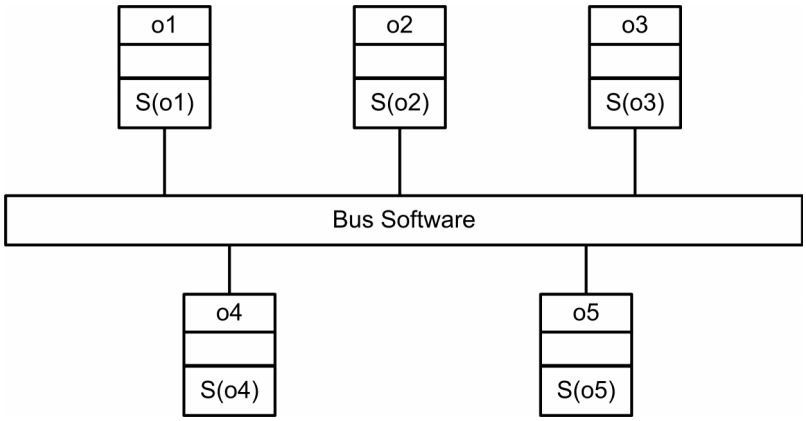


Figura 9.- Arquitectura de objetos distribuidos

Los objetos podrán distribuirse a través de varias computadoras en una red y comunicarse a través de un middleware denominado “intermediario de peticiones de objetos”, que proporcionará una interfaz transparente entre los objetos, ofertando un conjunto de servicios que permiten la comunicación entre objetos y que éstos sean añadidos y eliminados del sistema [102].

Entre las ventajas más significativas del paradigma de objetos distribuidos se podría afirmar que permite al diseñador del sistema retrasar decisiones sobre dónde y cómo deberían proporcionarse los servicios, es una arquitectura muy abierta que permite añadir nuevos recursos si es necesario. Esto es gracias al desarrollo de estándares de comunicación de objetos que permiten escribir objetos en diferentes lenguajes de programación, permitiendo comunicarse y proporcionar servicios entre ellos. Además, es un paradigma flexible y escalable, diferentes instancias del sistema pueden proporcionar los mismos servicios por objetos diferentes o reproducidos, para hacer frente a sobrecargas del sistema, pueden añadirse nuevos objetos a medida que la carga crece sin afectar al resto del sistema. Una última ventaja es que permite reconfigurar el sistema de forma dinámica mediante la migración de objetos a través de la red, pudiendo, por ejemplo, migrar un objeto que proporcione servicios al mismo procesador que los objetos que demandan los servicios, mejorando el rendimiento del sistema [102].

La principal desventaja de los sistemas de objetos distribuidos es que son mucho más complejos de diseñar que las arquitecturas cliente-servidor. Los sistemas cliente-servidor parece ser la forma más natural de concebir los sistemas. Éstos reflejan muchas interacciones humanas en las que las personas solicitan y reciben servicios de otras personas o máquinas especializados en proporcionar dichos servicios. Es más difícil pensar en una provisión de servicios generales [102].

Para la implementación de un sistema de objetos distribuidos será preciso, por lo tanto, de un middleware para la gestión de la comunicación entre objetos. Los objetos podrán implementarse utilizando diferentes lenguajes de programación, podrán ejecutarse sobre diferentes plataformas y sus nombres no precisan ser conocidos por el resto de objetos del sistema [102], [108], [109].

El middleware intermediario de peticiones de objetos deberá asegurar la transparencia de las comunicaciones de los objetos, lo que requerirá middleware a dos niveles: A nivel de comunicación lógica, proporcionando funcionalidades que permitan intercambiar datos y controlar la información sobre diferentes computadoras, con estándares tales como CORBA y COM, y a nivel de componentes, proporcionando una base para desarrollar componentes compatibles, gracias a estándares como CORBA, EJB o Active X, que proporcionan una base para la implementación de componentes con métodos estándar [102], [108].

Algunos ejemplos de implementación de sistemas de instrumentación que utilizan objetos distribuidos pueden encontrarse en [110], [111], [148].

2.5.3. Arquitecturas peer to peer

Un sistema peer to peer, también conocido como punto a punto o P2P, es un paradigma descentralizado en el cual los cálculos pueden llevarse a cabo en cualquier nodo de la red y donde no se realiza distinción entre clientes y servidores. En las aplicaciones P2P, el sistema se diseña con objeto de aprovechar la potencia computacional y disponibilidad de almacenamiento a través de una red de computadores potencialmente enorme. Tanto los estándares como los protocolos que permiten la comunicación a través de los nodos del sistema están embebidos en la propia aplicación, de tal manera que cada no debe ejecutar una copia de dicha aplicación [102].

Si bien es cierto que el paradigma de sistema distribuido P2P ha sido mayoritariamente utilizado en sistemas personales, caso de los sistemas de compartición de ficheros basados en los protocolos Gnutella y Kazza, utilizados para compartir ficheros sobre PCs de usuario, o los sistemas de mensajería instantánea tales como ICQ y Jabber, que proporcionan comunicaciones directas entre usuarios sin un servicio intermedio [149]. Cada vez es mayor el índice de utilización en empresas para que las redes de PCs soporten la potencia de dichas empresas [112]. Intel y Boeing, por ejemplo, han implementado sistemas P2P para aplicaciones que requieren computaciones intensivas. Para aplicaciones cooperativas que soportan trabajo distribuido, ésta parece ser la tecnología más efectiva [102].

Dentro del modelo P2P se pueden distinguir, a su vez, dos arquitecturas lógicas de red, una descentralizada y otra semicentralizada [102].

En principio, en el paradigma P2P cada nodo en la red podría conocer a cualquier otro nodo, conectarse con él e intercambiar datos. En la práctica, por supuesto, esto es imposible, ya que los nodos se organizan dentro de "localidades" con algunos nodos que actúan como puentes a otras localidades de nodos [102].

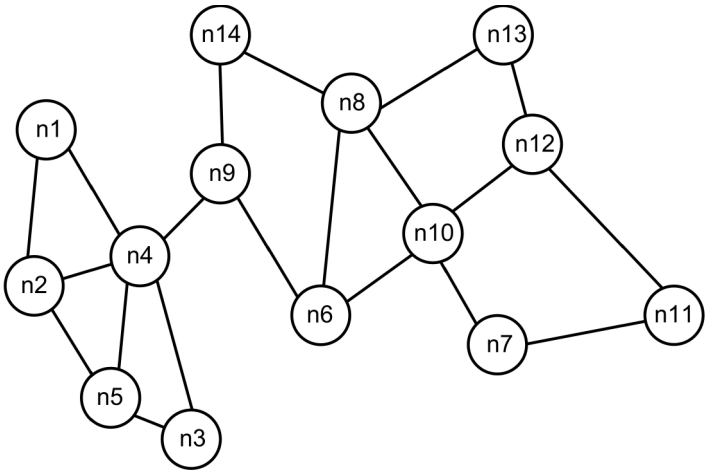


Figura 10.- Arquitectura peer-to-peer descentralizada

En una arquitectura P2P descentralizada, los nodos de la red no son sencillamente elementos funcionales, sino que además se comportan como "interruptores" de comunicaciones que posibilitan el encaminamiento de los datos y señales de control de un nodo a otro. En un sistema de compartición de archivos, por ejemplo, cada nodo tienen su propio almacén de archivos, no obstante, en el momento que se recupera un archivo por parte de un nodo, automáticamente se hace disponible a otros nodos. Cualquiera que precise un documento genera un comando de búsqueda que es enviado a los nodos de esa "localidad". Si algún nodo tiene el documento, lo devuelve a quien ha realizado la petición, en caso contrario, encaminan la búsqueda a otros nodos, de forma que si lo encuentra puede encaminarlo al nodo origen de la petición [102].

El paradigma P2P descentralizado tiene la ventaja de ser altamente redundante, y por lo tanto es tolerante a defectos y tolerante a nodos desconectados de la red. Sin embargo existen sobrecargas obvias en el sistema ya que la misma búsqueda puede ser procesada por muchos nodos diferentes y hay una sobrecarga significativa en comunicaciones entre iguales replicadas. Una alternativa a este problema lo constituye el paradigma P2P semicentralizado, en el cual uno o más nodos actúan como servidores para facilitar las comunicaciones entre los nodos.

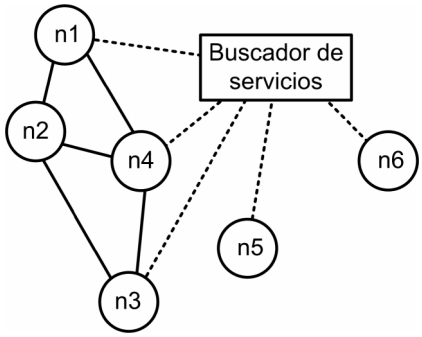


Figura 11.- Arquitecturas peer-to-peer semicentralizada

En una arquitectua semicentralizada, el papel del servidor es ayudar a establecer contactos entre iguales en la red o para coordinar los resultados de un cálculo. Un ejemplo típico de aplicación serían, por ejemplo, los servicios de mensajería instantánea. Una vez que éstos son encontrados, se pueden establecer comunicaciones directas y la conexión con el servidor es innecesaria.

2.5.4. Arquitecturas de sistemas orientados a servicios

El desarrollo de la WWW impulso que el acceso de máquinas cliente a servidores remotos fuera de sus propias organizaciones. Las organizaciones, convirtiendo la información en formato HTML hacían ésta accesible por las computadoras. No obstante, el acceso se llevaba a cabo únicamente a través de un navegador Web, y el acceso directo a almacenes de información por otros programas no era práctico.

Para dar solución a esta problemática se propuso el concepto de “servicio Web”. Mediante la utilización de un servicio Web, las distintas organizaciones que quieren hacer accesible la información a otros programas, lo podrán realizar definiendo y publicando una interfaz de un servicio Web. De forma genérica, un servicio Web podría definirse como una representación estándar para cualquier recurso computacional o de información que pueda ser usado por otros programas.

Un servicio Web es una instancia de una noción más genérica de un servicio, la cual se define en [103] como: “Un acto o realización ofertada por una de las partes a otra. Si bien el proceso puede estar asociado a un producto físico, la realización es esencialmente intangible, y no se convierte normalmente en propietaria de cualquiera de los factores de producción”.

La finalidad última de un servicio, por consiguiente, es que la provisión de servicio es totalmente independiente de la aplicación que usa dicho servicio. De esta manera, los proveedores de servicios pueden desarrollar servicios especializados y ofertados a un cierto número de usuarios de servicios desde diferentes organizaciones. Así, las aplicaciones pueden implementarse enlazando los servicios disponibles desde varios proveedores utilizando bien un lenguaje de programación estándar o bien un lenguaje de instrumentación de servicios.

Existen varios modelos de servicios, desde el modelo JINI [113], pasando por los servicios Web [114] y servicios de rejilla [115]. Todos estos modelos operan siguiendo el esquema descrito en la siguiente figura:

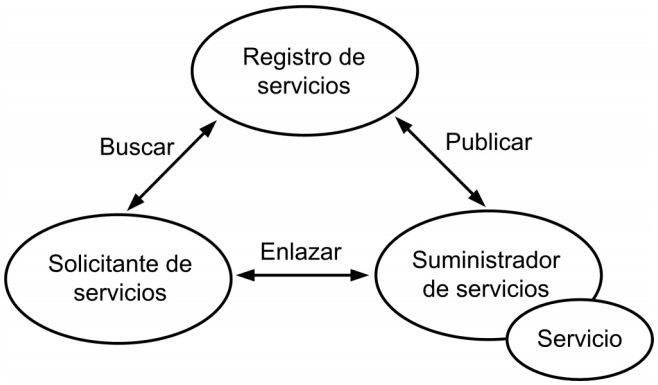


Figura 12.- Arquitectura conceptual de un sistema orientado a servicios

En el esquema mostrado anteriormente, un proveedor de servicio oferta un servicio definiendo su interfaz y definiendo la funcionalidad del servicio. Un usuario del servicio enlaza este servicio en su aplicación. Esto significa que la aplicación del usuario solicitante incluye código para llamar al servicio y procesa el resultado de la llamada al servicio. Para asegurar que el servicio puede ser accedido por usuarios externos a dicho servicio, el proveedor de servicios registra una entrada en el servicio de registro que incluye información sobre el servicio y lo que hace.

Dentro del marco de los servicios Web, podemos destacar tres estándares fundamentales que permiten la comunicación entre servicios Web: SOAP, WSDL y UDDI. En primer lugar, SOAP (Simple Object Access Protocol) define una organización para intercambio de datos estructurados entre servicios Web. En segundo lugar, WSDL (Web Services Description Language) define cómo pueden representarse las interfaces de servicios Web. Por último, UDDI (Universal Description, Discovery and Integration), es un estándar de búsqueda que define cómo puede organizarse la información de descripción de servicios, usada por los solicitantes de los servicios para encontrar los servicios. Todos ellos se basan en XML, un lenguaje legible por los humanos y las máquinas.

2.6. Ámbitos de aplicación de la instrumentación virtual remota

El sector de la instrumentación virtual remota, desde sus orígenes en los años 70, ha sufrido una enorme evolución. Este desarrollo se ha visto impulsado y promovido por un conjunto de necesidades aparecidas en la comunidad industrial, científica y educativa, fundamentalmente.

La instrumentación virtual permite a las industrias y a los laboratorios realizar medidas automatizadas en aplicaciones simples o complejas con una alta flexibilidad y adaptabilidad. Así, es posible modificar el procedimiento de medida o de control cambiando simplemente el algoritmo ejecutado en el computador sin necesidad de reemplazar los componentes de hardware. Estas prestaciones permiten que la actividad experimental se desarrolle de forma rápida y sencilla, especialmente, cuando el procedimiento de medida no está

completamente definido. Además, la reusabilidad y generalidad de los componentes de media programables reducen los costes de adquisición y operación. Por tanto, la tendencia actual en la implementación de sistemas de medida y de control se orienta hacia el uso de estas tecnologías.

En algunos casos, el elevado coste de determinados sistemas de instrumentación dificulta la adquisición de los mismos por determinadas comunidades que podrían estar interesadas en su estudio y manejo. La implementación de una arquitectura de acceso remoto podría posibilitar el acceso a terceras comunidades de este tipo de instrumentación, creando nuevos modelos de negocio, potenciando la colaboración entre instituciones y promoviendo la globalización del conocimiento [150].

Otro hecho donde la instrumentación remota tiene lugar surge de la necesidad de disponer de un equipamiento de instrumentación de forma puntual y en un breve espacio de tiempo. Tal puede ser el caso de sistemas de calibración o de medición de parámetros de calidad.

Unido a lo anterior, también existen circunstancias en las que se precisa la necesidad imperiosa de realizar a distancia el control y monitorización de ciertos procesos dada las características propias de la experimentación, tal es el caso de las centrales nucleares, por ejemplo [60].

Alguna instrumentación únicamente puede tener lugar en una localización en concreto, caso de los radiotelecopios, donde se precisa de una infraestructura de compartición de sistemas e información [60].

El factor humano también está presente en el ámbito de las necesidades explícitas de sistemas de instrumentación remota, como es en aplicaciones de telemedicina, en las que se requiera de un especialista situado a cientos de kilómetros, por ejemplo, para la realización de intervenciones quirúrgicas, etc. O bien la formación a distancia, una de las aplicaciones más extendidas de los sistemas de instrumentación virtual remotos, dando lugar al subgrupo conocido como laboratorios de instrumentación virtual remoto o simplemente laboratorios virtuales.

Toda esta casuística ha dado lugar a la necesidad de desarrollar nuevas arquitecturas de sistemas de instrumentación virtual remotos, posibilitando la equidad de oportunidades, la unificación y la globalización de los sectores productivos, científicos y educativos.

Por tanto, podemos afirmar que el diseño de nuevas técnicas de instrumentación virtual que permitan el acceso remoto y compartido a la instrumentación científica, educativa e industrial abre nuevas oportunidades para la industria, la ciencia, la educación y los negocios [150].

De un estudio realizado durante el desarrollo de la presente investigación sobre un total de 140 artículos de instrumentación virtual de revistas indexadas por el JCR de los últimos años, dividiendo los sectores de aplicación de la misma en industria, ciencia y educación, se obtienen los siguientes resultados:

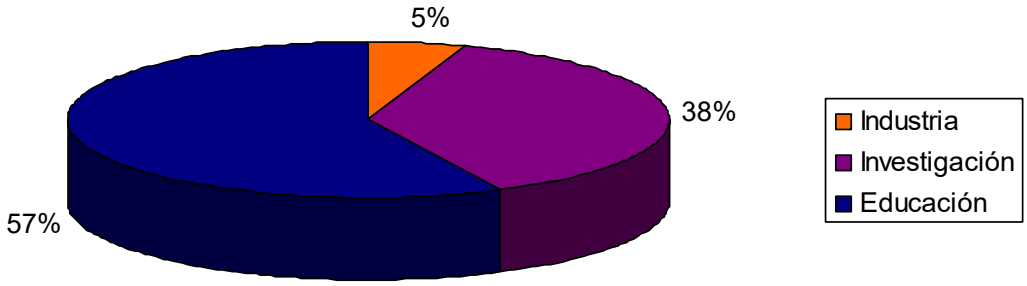


Figura 13.- Sectores de utilización de los sistemas de instrumentación virtual

De la anterior figura se puede observar que uno de los sectores donde la instrumentación virtual tiene mayor cabida es el sector académico, abarcando un 57% de las aplicaciones descritas en la bibliografía. Esto se debe precisamente a algunos de los argumentos anteriormente mencionados. Coste, distancia, uso puntual y deslocalización de recursos. Por otra parte, seguido por un 38% nos encontramos al sector de la investigación como segundo usuario de los sistemas de instrumentación. En ambos casos el uso de la instrumentación virtual remota tiende en su mayoría al subconjunto de los laboratorios de instrumentación virtual, tal y como se ha citado anteriormente. El último valor, un 5%, corresponde a la industria, aunque probablemente dicho porcentaje sea mayor dado que la fuente de localización de aplicaciones tiene un ámbito educativo-científico claramente definido.

Otra clasificación de los ámbitos de utilización de la instrumentación virtual remota podría realizarse alrededor de las disciplinas que abarca. Si bien otros campos de investigación tienen su foco de interés muy localizado, no sucede así con la instrumentación virtual, teniendo cabida en aplicaciones que abarcan desde la física hasta la robótica. Algunas de las disciplinas donde la instrumentación remota tiene mayor aceptación y uso son la electrónica y la robótica [151], seguidas de la automática y física [151]. A continuación se presenta un gráfico con los resultados del estudio de un total de 41 publicaciones en [151].

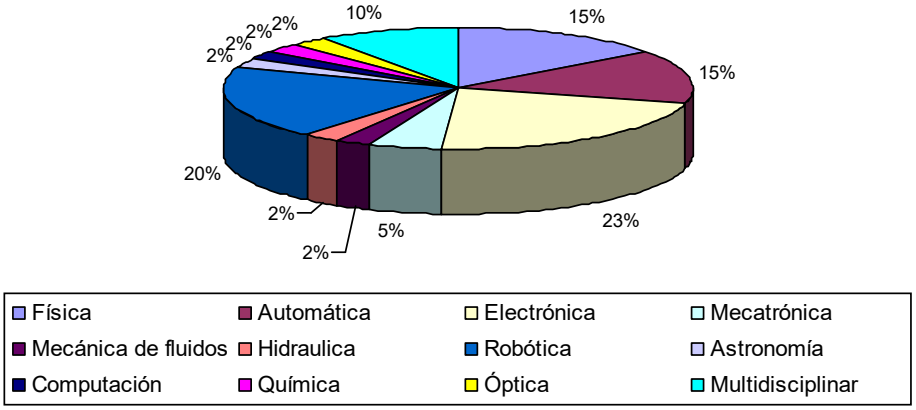


Figura 14.- Empleo de los sistemas de instrumentación en base a la disciplina

3. Bancada de referencia para la experimentación

Para poder poner de manifiesto alguno de los aspectos presentados en el presente trabajo de investigación resulta preciso diseñar una bancada de pruebas para llevar a cabo la experimentación. Si bien es cierto que existen en la actualidad infinidad de modelos y arquitecturas de instrumentación aplicados a diferentes disciplinas y materias, en apariencia totalmente distintas, lo cierto es que los sistemas de instrumentación, especialmente los “laboratorios de instrumentación remota”, presentan más similitudes que diferencias.

A continuación se presenta una breve descripción de la bancada empleada para abordar la temática presentada en la presente tesis. Dicha bancada refleja un puesto de instrumentación clásico, con la peculiaridad de incluir un sistema procesador basado en un DSP de la familia Texas Instruments, dado en ingente desarrollo que este tipo de plataformas está teniendo en la actualidad y su inclusión en sistemas de instrumentación como parte del DUT.

El puesto del laboratorio de instrumentación remota estará formado por 3 partes claramente diferenciadas: El equipamiento de instrumentación, una placa de desarrollo del sistema DSP y un ordenador personal que actuará como interfaz con el cliente del mismo y los sistemas de instrumentación implicados.

Figura 15.- Fotografía del puesto de trabajo de la bancada

3.1. Sistema de desarrollo basado en el DSK6711

Se empleará un DSK6711 de la firma Spectrum Digital que se basa en la arquitectura DSP TMS320C6711 de Texas Instruments. Dicho sistema emplea el puerto de comunicación paralelo para la comunicación con el PC, permitiendo el desarrollo de programas de aplicación que son cargados en la

placa y ejecutados y depurados. La comunicación con el PC se realiza a mediante el empleo del software de aplicación Code Composer Studio.

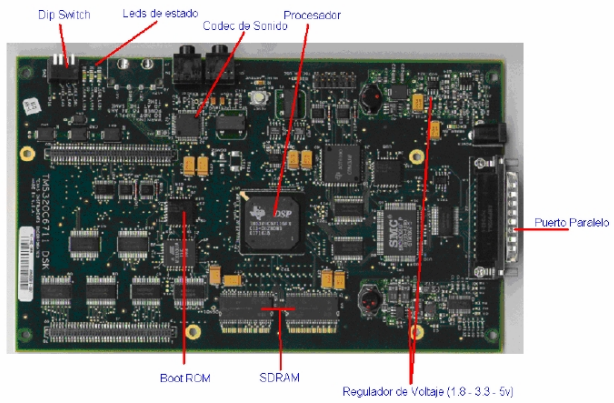


Figura 16.- Sistema de desarrollo DSK basado en DSP de Texas Instruments

El DSK dispone de varios periféricos de entrada-salida disponibles; siendo empleados concretamente dos conectores Jack correspondientes a una señal de entrada analógica y una señal de salida analógica del DSP. La entrada y salida se conectarán a los equipos de instrumentación. En la entrada se introducirá una señal producida por un generador de funciones. Dicha señal será procesada por el programa que previamente haya sido volcado sobre el sistema de desarrollo y se obtendrá una señal de salida que será conectada a la entrada de un osciloscopio.

3.2. Instrumentación

En relación a los equipos de instrumentación propiamente dichos, se han empleado tres de los dispositivos que encontramos con mayor frecuencia en los laboratorios de instrumentación, un generador de funciones Agilent 3320A, una fuente de alimentación Agilent E3631A y un osciloscopio digital Agilent DSO3062A.

Figura 17.- Fotografía de los equipos de instrumentación empleados

Todos estos dispositivos disponen de interfaz de comunicación GPIB, la cual será empleada para la conexión con el ordenador personal.

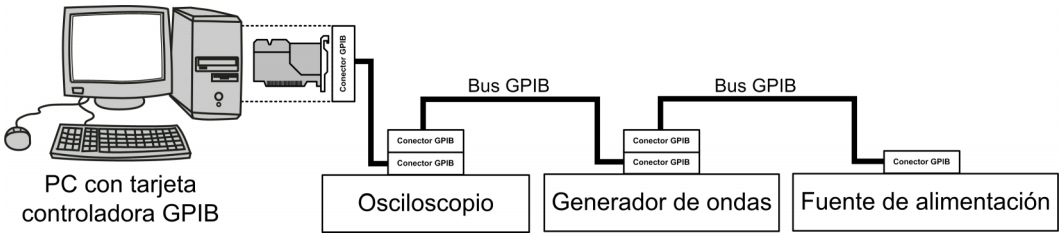


Figura 18.- Representación del conexionado de la instrumentación al bus GPIB

La salida del generador de funciones será conectada a una entrada analógica del sistema DSK (códec de sonido) y a la entrada de uno de los canales del oscloscopio. La tarjeta de desarrollo del DSP es alimentada por la fuente de alimentación Agilent E3631A, controlada también a través del bus GPIB. Finalmente, una salida analógica del DSK es conectada al otro canal de entrada del osciloscopio, pudiendo monitorizar en el mismo la señal generada por el generador de funciones y la disponible a la salida del DSP[3].

3.3. Ordenador personal

El ordenador constituye una de las partes más importantes del proyecto, puesto que en éste residen todas las aplicaciones encargas de la gestión tanto de la interfaz con el usuario remoto como la comunicación con los distintos equipos de instrumentación. Para establecer la comunicación con la instrumentación de medida, se ha dotado al mismo de una tarjeta GPIB de comunicación.

El esquema general de conexionado de la bancada experimental es como se detalla en la Figura 19.

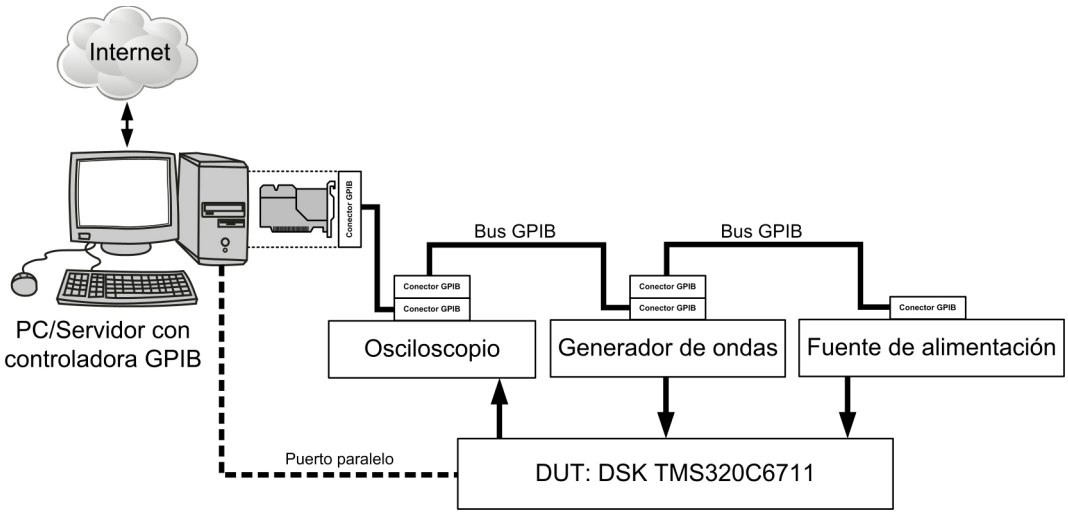


Figura 19.- Conexionado de los diferentes elementos de la bancada experimental

[3] Uno de los experimentos desarrollados es la implementación de técnicas de implementación de filtros digitales en el DSP, pudiendo observar el efecto de los mismos comparando entrada y salida al DSP en el osciloscopio en función de la frecuencia.

4. eDSPlab - Laboratorio de referencia basado en LabVIEW

Uno de los paquetes software más utilizados en la actualidad en el diseño de sistemas o laboratorios de instrumentación remota es LabVIEW [10]. Podemos encontrar numerosos trabajos donde se describen implementaciones aplicadas a multitud de disciplinas de la ciencia y la tecnología, como son los desarrollados en [116] y [117], entre otros.

LabVIEW es un software propietario de la firma National Instruments. Entre sus bondades destacan su facilidad de uso, permitiendo a personas con escasos o nulos conocimientos de programación, el desarrollo de programas de una relativa complejidad. Esto es posible gracias a que emplea un lenguaje gráfico que sigue una metodología modular de división del programa en bloques donde las aplicaciones se diseñan siguiendo un diagrama de flujo que es el que controla la secuencia de ejecución.

Otra de las ventajas del uso de LabVIEW en los sistemas de instrumentación es que posee una potente galería de controles gráficos que permiten crear interfaces de usuario con asombrosa rapidez, los cuales son mapeados con los datos de programa.

Fruto del gran desarrollo de los sistemas de instrumentación, la empresa National Instruments pone a disposición del programador diversos “toolkits” que facilitan la integración de subsistemas, tal es el caso del Toolkit para DSPs de Texas Instruments, gracias al cual es posible el manejo del software de aplicación Code Composer Studio [118] y, por ende, los dispositivos DSP.

Además, presenta una elevada cantidad de funciones y servicios ya diseñados, como la disponibilidad de un servidor Web propio y la posibilidad de publicar los paneles remotos de los sistemas de instrumentación con apenas unos pequeños pasos de configuración [10], [15].

No obstante, la utilización de aplicaciones de alto nivel también presentan limitaciones y desventajas en comparación con otras soluciones.

El paquete software LabVIEW ha sido empleado para el diseño e implementación de un laboratorio de instrumentación remoto que responda a las necesidades de gestión, manejo y control de la bancada de instrumentación anteriormente descrita; con objeto de poner de manifiesto algunos aspectos relevantes sobre las virtudes e inconvenientes en el uso de este tipo de plataformas.

El diseño ha partido de un primer prototipo en el que el diseño LabVIEW únicamente se accedía a un conjunto de sistemas de instrumentación básica: Una fuente de alimentación, un generador de funciones y un osciloscopio [119]. Una vez comprobada la funcionalidad y efectividad técnica del sistema de control, se comenzó a estudiar y desarrollar la integración en el laboratorio remoto de un DSK de la familia TMS320C3x [120], al mismo tiempo que se analizó su integración dentro del contexto educativo, junto con otras herramientas multimedia desarrolladas en el seno del portal Web de una

asignatura de la titulación de Ingeniería de Telecomunicación, “Complementos de Sistemas Electrónicos Digitales”, empleando la técnica de mapas conceptuales descrita en [121] y [122] para adecuar la metodología de enseñanza. Paralelamente se comenzó a aplicar un modelo de aceptación tecnológica (TAM) para no sólo medir el uso de la herramienta sino obtener también las variables externas con una influencia significativa en su uso, para poder planificar futuras mejoras de la misma [123]. De lo anterior se extrajeron algunas conclusiones relativas a la necesidad de mejorar aspectos relativos al laboratorio de instrumentación remoto implementado, como la necesidad de mejorar la interfaz y la forma en que los usuarios interactuaban con el sistema; decidiéndose incluir un nuevo sistema de desarrollo basado en el DSK TMS320C6000 que, por sus diferencias respecto al sistema anterior implicarían una importante modificación de la implementación anterior [124].

4.1. Descripción general del sistema

En relación a la solución técnica desarrollada, por una parte, será preciso crear una interfaz de comunicación entre el ordenador personal y los sistemas de instrumentación, incluyendo al DSK, de tal manera que sea posible acceder a través de ellos. Por otra parte, se deberá proveer de mecanismos que hagan accesible el panel de control de los equipos de instrumentación a usuarios remotos conectados a través de Internet.

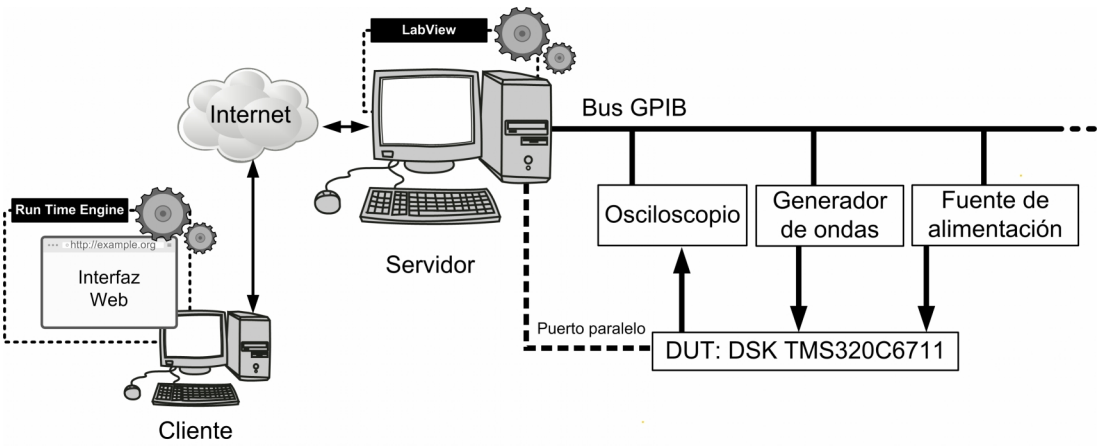


Figura 20.- Sistema de instrumentación remota implementada en LabVIEW

Emplearemos LabVIEW para crear un “Instrumento Virtual o VI” (VI, del acrónimo Virtual Instrument) que simulará los principales elementos de control de los equipos de instrumentación (botones, ruedas, etc.) en la pantalla del PC y los cuales serán empleados para comunicarnos con la instrumentación, del mismo modo que si estuviéramos controlándola físicamente. Finalmente, estableceremos los mecanismos necesarios para proporcionar la publicación del instrumento virtual a clientes remotos a través de Internet, es decir, haremos accesible el sistema de instrumentación a través de Internet, discutiendo aquellos aspectos más relevantes.

Disponemos, por tanto, de dos partes claramente diferencias en el diseño de un VI, por una parte el “panel frontal”, que es la interfaz que se presenta al usuario de la aplicación de control, bien local (en el propio servidor)

o remotamente y donde se ubican los controles de entrada (botones, interruptores, etc.) y de salida (gráficas, indicadores de estado, etc.).

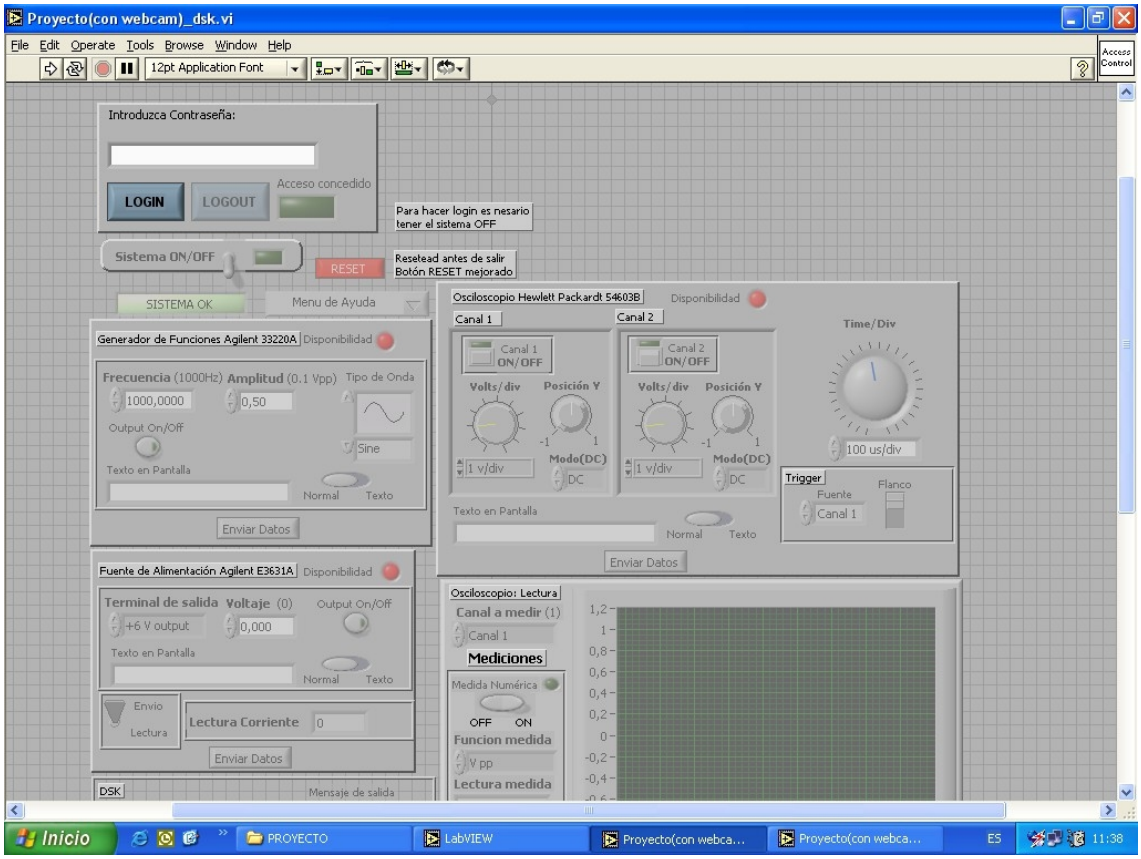


Figura 21.- Ejemplo de panel frontal en LabVIEW

Por otra, nos encontramos con la ventana de diseño donde se programa la funcionalidad del sistema en forma de diagrama de bloques.

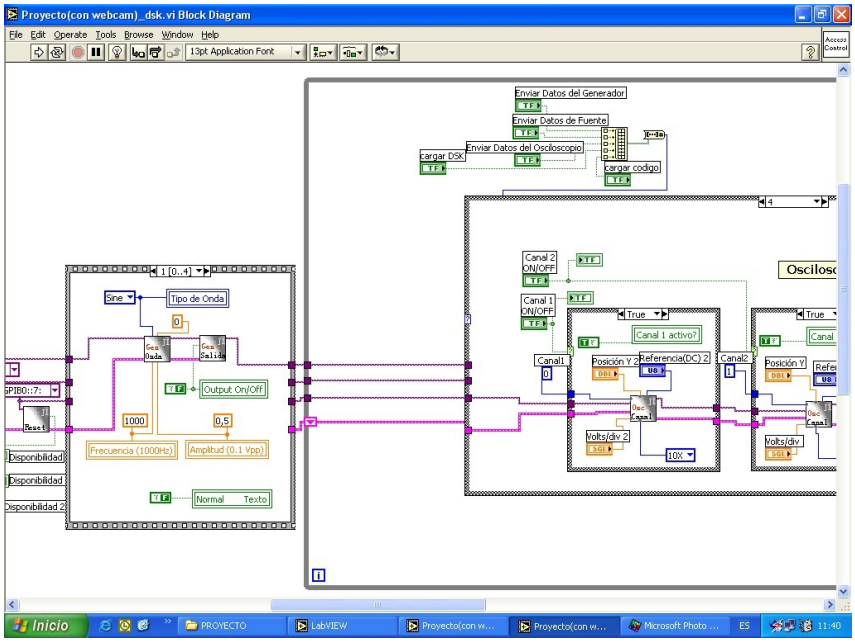


Figura 22.- Ejemplo de ventana de programación en LabVIEW

4.2. Módulos del laboratorio de instrumentación eDSPlab

El diseño del instrumento virtual que se presentará al usuario para el manejo de la instrumentación electrónica se ha estructurado en *frames*; de tal forma que se realiza una ejecución de los distintos fragmentos o *frames* de código secuencialmente. A cada *frame* se le ha asociado un conjunto de funciones en particular, así, el primer *frame* es el encargado de la inicialización de las variables del sistema, por ejemplo.

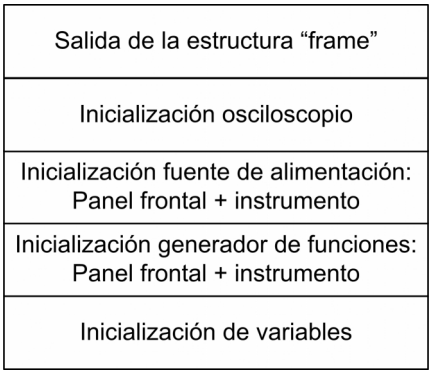


Figura 23.- Estructura de los frames del instrumento virtual eDSPlab

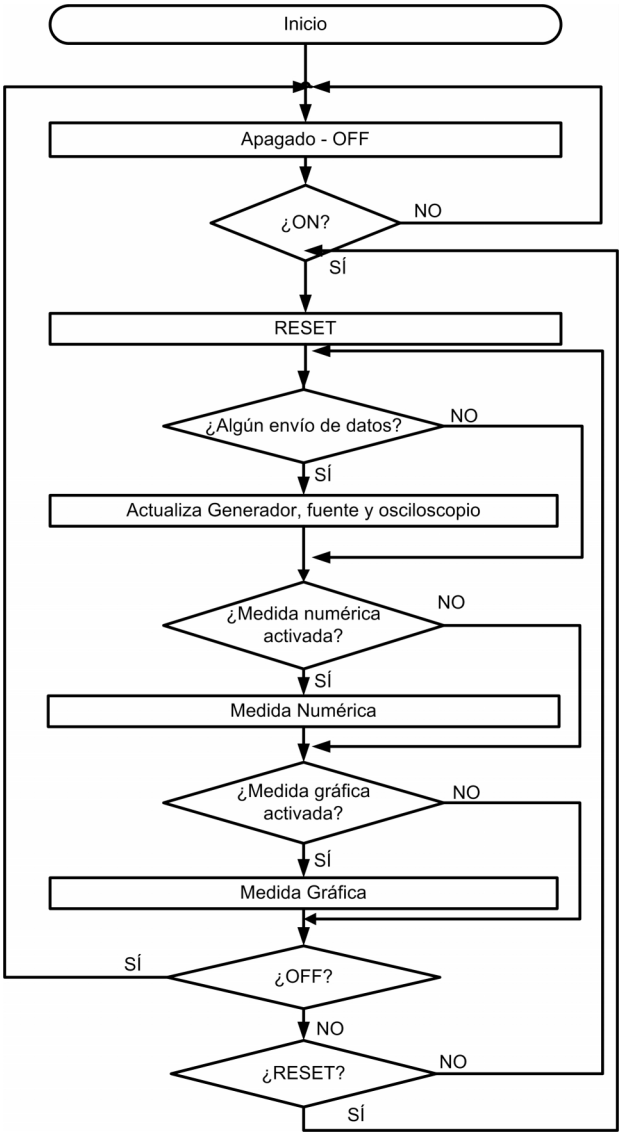


Figura 24.- Diagrama de flujo del bloque principal del sistema de instrumentación

LabVIEW dispone de una potente librería de controles cuya apariencia imitan a los que podemos encontrar en los instrumentos reales, de tal forma que resulta sencillo implementar un interfaz gráfico de forma rápida y que refleje lo más fielmente posible los paneles de control de la instrumentación real [10].

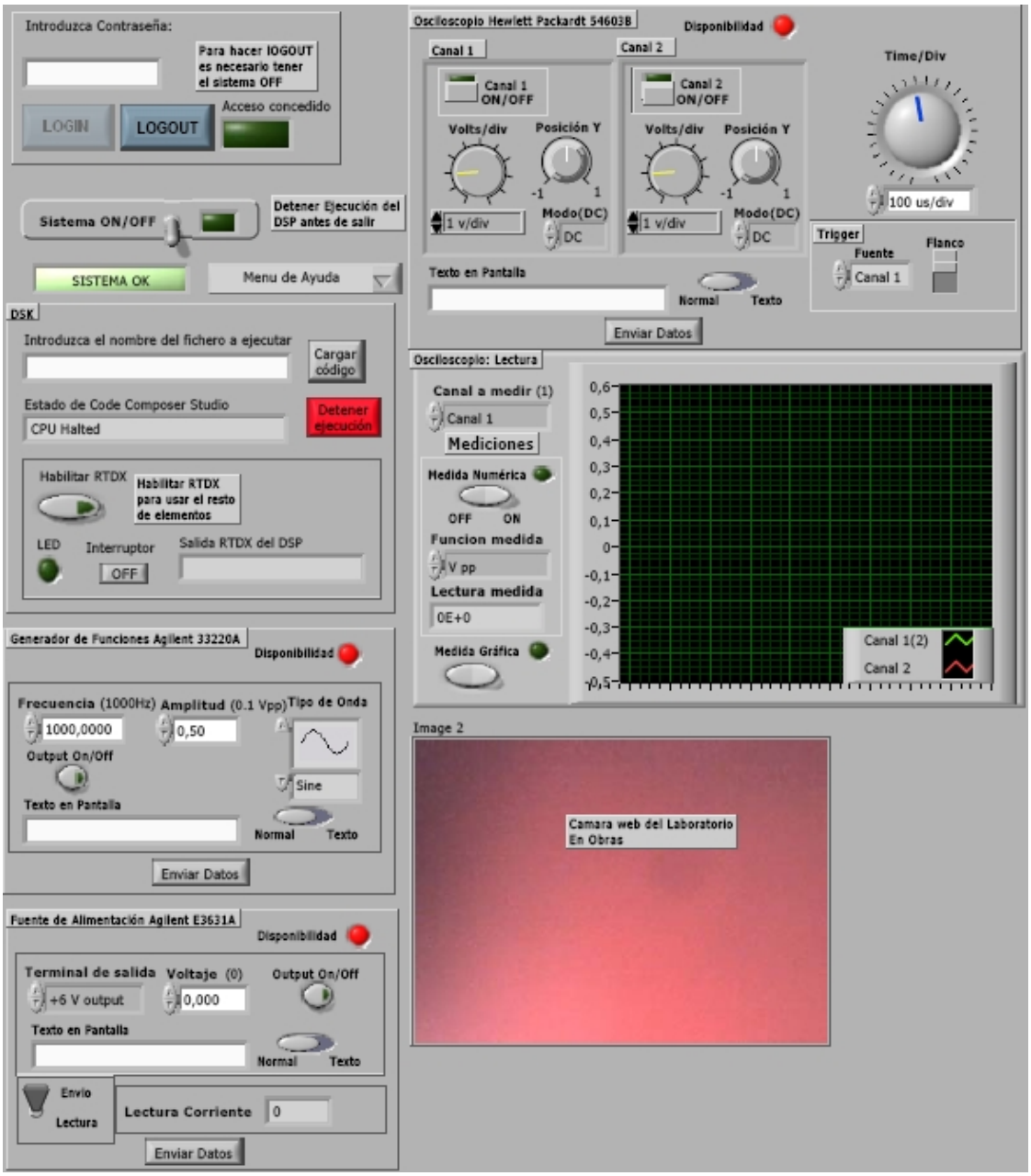


Figura 25.- Panel frontal que se ha implementado como interfaz gráfico

Se ha diseñado un panel frontal dividido en cinco diferentes secciones o módulos, correspondientes al generador de funciones, la fuente de alimentación, el osciloscopio, la interfaz con el DSK, etc. A continuación se describe brevemente la funcionalidad de cada uno de éstos.

Módulo de autentificación

Se ha dotado al laboratorio de un sistema de autentificación para poder controlar el manejo del mismo. Se basa en la utilización de contraseñas; de tal forma que el sistema de instrumentación queda inhabilitado mientras no se proporcione la misma.

Módulo de información del sistema

Este módulo será el encargado de la activación/descativación del sistema empleando un control gráfico. Además proveerá del menú de ayuda y un indicador de estado del sistema.

Módulo de manejo y comunicación con el DSK

En este módulo se gestiona el procedimiento de carga del fichero o programa diseñado para el DSP en la placa de desarrollo. En la sección del panel frontal destinado a tal fin se deberá proporcionar el nombre y extensión del mismo, el cual habrá sido previamente cargado en la máquina servidora. También dispone de un control que nos informa del estado del programa Code Composer Studio. A través del submódulo RTDX[4] (Real Time Data Exchange) [125] implementado en este módulo, también es posible establecer un canal de comunicación que permita interactuar con el DSP (A través de un interruptor o mensajes de salida, por ejemplo).

Módulo del generador de funciones

Dicho módulo permite la activación/desactivación del generador de funciones y el control de las principales funcionalidades del mismo (selección del tipo de onda, frecuencia, amplitud, etc.) empleando la interfaz GPIB.

Módulo de la fuente de alimentación

Del mismo modo que el módulo del generador, se permite la activación/desactivación del control de la fuente y el manejo de sus principales funcionalidades.

Módulo del osciloscopio

Este módulo es el encargado del control del osciloscopio. Se distinguen dos subpaneles, uno enfocado a la configuración del mismo (activacion/desactivación de canales, modos de funcionamiento AC/DC, etc.), y un segundo subpanel dedicado a la representación de la información y la realización de medidas tanto gráficas como numéricas.

4.3. Comunicación con el DSK y gestión de ficheros

Para el manejo de los DSP de la familia C6000 de Texas Instruments con el entorno LabVIEW existen dos posibles alternativas. En primer lugar es posible utilizar LabVIEW junto con un módulo conocido como “NI LabVIEW DSP[5]” [126], el cual nos permite diseñar de manera gráfica en el propio entorno LabVIEW aplicaciones para los DSP, sin utilizar el entorno Code Composer Studio y haciendo tanto el software como el hardware “invisibles” al programador.

[4] RTDX (Real Time Data Exchange) es una funcionalidad que proporciona Texas Instruments mediante la cual es posible transmitir y recibir datos entre un equipo host, como podría ser un ordenador personal, y una placa DSP, sin que ello afecte a la aplicación que en ese momento esté ejecutándose en la misma.
[5] En la actualidad este producto está discontinuado.

En segundo lugar, se puede emplear LabVIEW junto con el "DSP Test Integration Toolkit[6]" de tal forma que podamos interactuar desde el entorno LabVIEW con el programa Code Composer Studio.

En la implementación de eDSPlab se ha optado por la segunda opción, pues uno de los objetivos es estudiar las capacidades del entorno y compararlas con una nueva propuesta, presentando mayor problemática la solución adoptada, como veremos más adelante.

Por tanto, tal y como hemos comentado anteriormente, el módulo que gestiona el DSK dispone de dos submódulos claramente diferenciados, uno dedicado a la gestión del fichero asociado al proyecto o programa que deseamos que se ejecute en el DSP y un segundo submódulo encargado de gestionar la funcionalidad RTDX que nos permite el intercambio de información entre servidor y el DSP sin necesidad de parar la ejecución del mismo.

4.3.1. Gestión de ficheros

Para poder ejecutar un programa (o proyecto) en el DSP de forma remota será necesario realizar varios pasos: En primer lugar, subir el fichero al servidor, seguidamente cargarlo en la placa del DSK y, finalmente, ejecutarlo.

El primero de los pasos implicados, la subida del fichero al servidor, no resulta una operación trivial. En versiones anteriores de los DSP el fichero que se volcaba para su ejecución era del tipo .dsk, tratándose de un fichero de texto [120]. Esto posibilitaba la utilización de una ventana en la cual podía "pegarse" directamente el contenido del fichero a ejecutar y que, al pulsar un botón del panel, dicho contenido se almacenara en un archivo en el servidor y pudiera ser volcado al DSP. Sin embargo, en las versiones superiores de los DSP y otras arquitecturas esto no es posible, ya que en muchas ocasiones los ficheros volcados son de tipo binario, tal es el caso de los ficheros asociados al DSP empleado en la bancada, cuya extensión es de tipo ".out". Frente a este problema se plantean varias posibles soluciones.

Por una parte, existe la posibilidad de emplear un Toolkit proporcionado por LabVIEW, conocido como "Internet Connectivity Toolkit", que nos permite incorporar dentro de la interfaz Web del laboratorio remoto un cliente para la transferencia de ficheros, utilizando el protocolo de transferencias de ficheros FTP o un formulario PHP, tendiendo como mayor ventaja la posibilidad de tener integrado en el propio entorno toda la funcionalidad del sistema. Sin embargo, esto tiene un coste adicional elevado[7].

Otra forma de solucionar el problema es desarrollar una interfaz Web paralela en un lenguaje de programación que se ejecute en el servidor, como PHP, que nos permita, a través de un formulario, la transferencia de un fichero .out a un directorio del servidor. La ventaja de utilizar esta solución es que el lenguaje PHP es empleado por numerosos sistemas CMS y LMS, como es el caso de Moodle, que están teniendo una gran acogida en diversos sectores,

[6] En la actualidad este Toolkit se encuentra integrado en el entorno LabVIEW.

[7] El coste adicional del Internet Connectivity Toolkit para la versión LabVIEW empleada en el presente trabajo de investigación es de unos 500 dólares. Actualmente las nuevas versiones de LabVIEW incorporan esta funcionalidad dentro de sus distribuciones.

especialmente el académico, pudiendo plantearse la integración del laboratorio en un entorno de enseñanza virtual Moodle [127], [128] e incluirlo en una plataforma conjunta con otras herramientas u objetos de aprendizajes desarrollados, como herramientas multimedia [129], [130], [131], [132] entornos de laboratorio VRML-3D [134], [135]. Además, el uso del lenguaje PHP no conlleva ningún coste adicional. Sin embargo, esta solución tiene varias implicaciones desventajosas, como que el sistema de subida de archivos es totalmente ajeno al entorno LabVIEW, se pueden presentar problemas de seguridad, el usuario debe emplear dos interfaces para la gestión de los ficheros, y se precisa utilizar dos puertos diferentes para atender las peticiones de los clientes al servidor, uno asociado al servidor LabVIEW y otro asociado al servidor de gestión de subida de ficheros.

En la implementación de eDSPlab se ha optado por la solución basada en el empleo de un formulario PHP. Así pues, el usuario remoto deberá emplear el mismo para cargar el proyecto .out en el servidor y, posteriormente, indicar al panel frontal de LabVIEW el nombre del fichero correspondiente al proyecto que se desea cargar en el DSK.

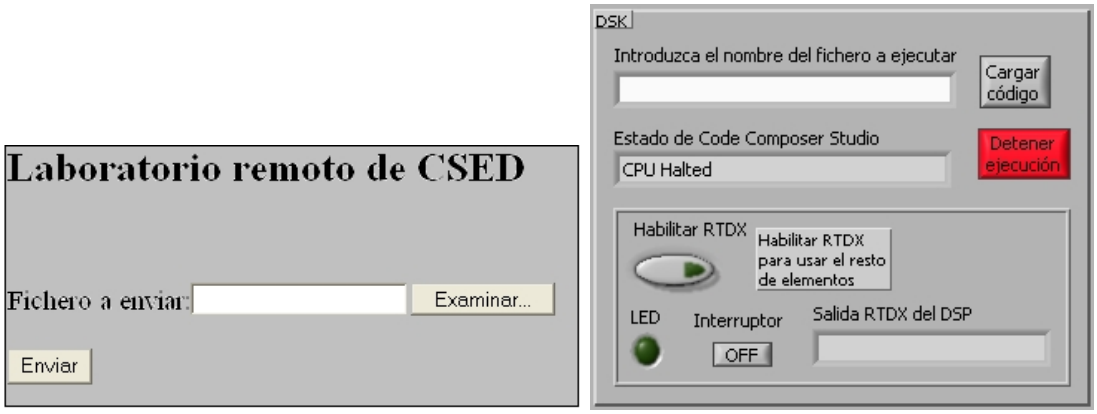


Figura 26.- Izquierda: Interfaz para la subida de archivos al servidor. Derecha:

4.3.2. Carga del programa en el DSK y gestión de RTDX

Una vez validado el fichero .out subido, gracias al empleo del "DSP Test Integration Toolkit[8]" del LabVIEW, podemos interactuar con el programa Code Composer Studio para realizar los procesos de reseteo del sistema de desarrollo DSK, la carga del fichero .out transferido por el usuario remoto y la ejecución del mismo en el DSP [152].

También es posible definir canales RTDX de comunicación entre el DSP y el panel frontal, en particular, se ha implementado dos canales de salida asociados a un control tipo "piloto o LED" en el panel frontal y una ventana para recibir mensajes de texto, así como un canal de entrada asociado a un control tipo "interruptor".

[8] En la actualidad este Toolkit se encuentra integrado en el entorno LabVIEW.

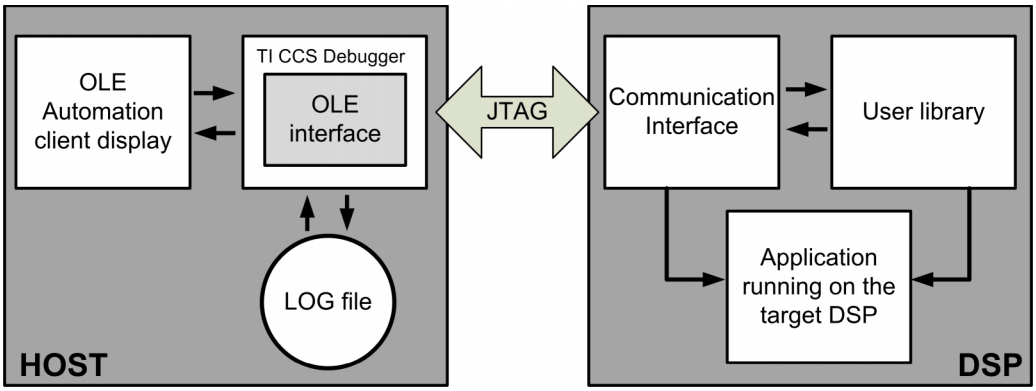


Figura 27.- Representación del flujo de datos en RTDX

4.4. Acceso remoto de clientes en LabVIEW

Para acceder al sistema de instrumentación localmente, el usuario únicamente precisará que en la máquina servidora esté activo el paquete LabVIEW y el instrumento virtual diseñado ejecutándose. A través del panel frontal del mismo podrá interactuar con la instrumentación accionando o capturando eventos.

Cuando se desea acceder desde otro dispositivo remoto al panel frontal, a través de Internet, resulta preciso, en primer lugar, que la máquina donde se ejecuta el programa tenga conectividad a Internet y una dirección IP visible. LabVIEW dispone de un servidor Web que nos ofrece la posibilidad de hacer accesible el panel frontal en otro ordenador remotamente sin más que activarlo en la máquina servidora y configurar los datos de acceso. Para ello, existen dos opciones basadas en los modelos de cliente fino y cliente grueso [119].

Una primera alternativa de conexionado es la que se basa en que el cliente o usuario remoto tenga también instalado el programa LabVIEW en su máquina, accediendo al instrumento remoto mediante una opción “Connect to remote Pannel”, indicando la dirección IP del servidor y el nombre de la aplicación o instrumento virtual (con extensión .vi) y solicitando el control del mismo. Este método nos permite tener un control total del panel frontal desde la máquina cliente, del mismo modo que si estuviera accediendo desde el servidor.

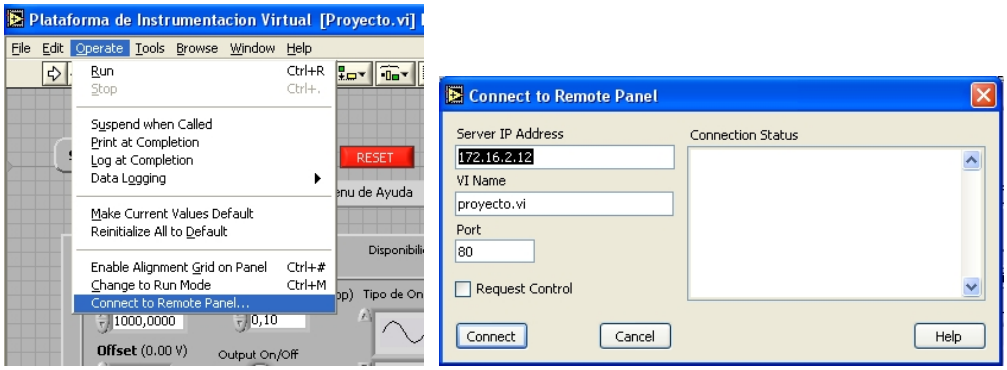


Figura 28.- Acceso al panel remoto desde el cliente

Sin embargo, el anterior método, basado en el modelo de cliente grueso [119], donde existe una aplicación en el usuario remoto encargada de gran parte del proceso de acceso remoto y control, presenta varios inconvenientes, entre los que destacan la necesidad de una licencia por cada cliente, la sobrecarga de la CPU y memoria de las máquinas cliente y las limitaciones en cuanto al tipo de dispositivo y sistemas operativos que pueden acceder al sistema de instrumentación remota.

Una segunda opción, más interesante, que no precisa la instalación de LabVIEW en la máquina cliente, se basa en la simple utilización de un navegador Web en el usuario remoto, convirtiéndolo en un cliente delgado. Para que esto sea posible es preciso que en el extremo servidor se genere un documento legible por el navegador del cliente.

LabVIEW dispone de un editor HTML que permite hacer visible un instrumento virtual de forma remota en un cliente mediante el empleo de un simple navegador y un plug-in que precisa ser instalado previamente en la máquina cliente. El editor HTML de LabVIEW nos ofrece varias opciones a la hora de crear el documento[9], pudiendo elegir entre la posibilidad de ver el panel como una imagen estática, como una imagen con refresco o disponiendo del control total del panel remoto. Las dos primeras opciones no son adecuadas puesto que no ofrecen la posibilidad de interactuar con el panel al mismo tiempo que observar los cambios producidos en el mismo.

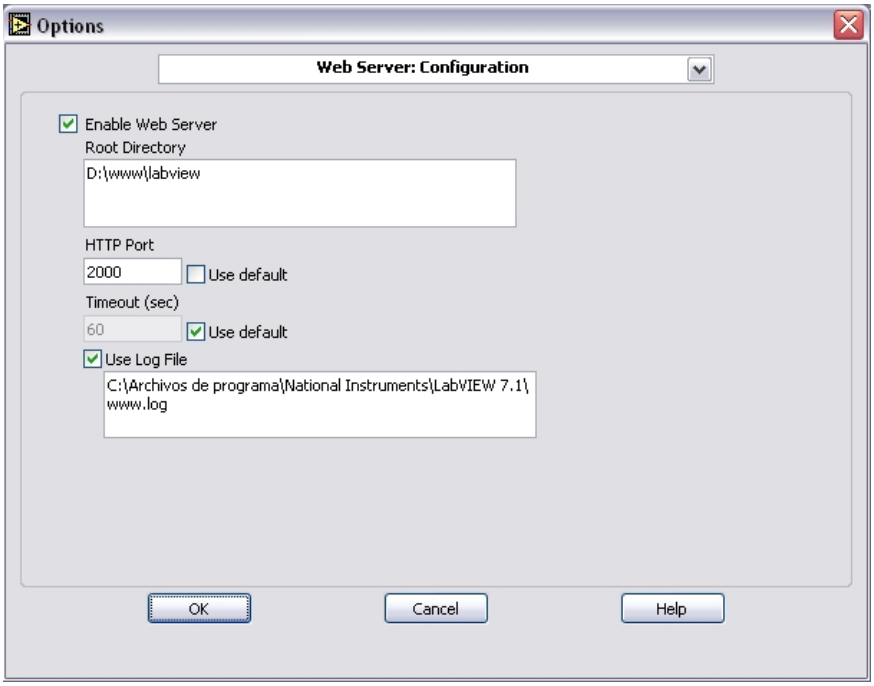


Figura 29.- Ejemplo de configuración del Web Server

Para tener accesible la página HTML generada por LabVIEW es preciso, por un lado, activar el servidor Web[10] e indicar al mismo la ruta donde se ubicará el fichero generado y, por otra, se debe activar la licencia "Remote

[9] Accesible desde la opción WebPublishing Tool del menú Tools de LabVIEW.
[10] Accesible desde la pestaña Web Server Configuration desde Options en el menú Tools.

Panel License", la cual debe ser solicitada a National Instruments y únicamente es válida para la máquina donde se solicita.

Dado que el puerto 80 es empleado por el servidor Apache que se emplea para la gestión de la página PHP de subida de ficheros, es preciso indicarle al servidor la utilización de un puerto diferente, en nuestro caso hemos empleado el puerto 2000, de tal manera que en el navegador del cliente será preciso indicar el puerto por el que estamos accediendo al servidor (por ejemplo: http://dominio_intrumento_virtual:2000/nombre_archivo_html.htm).

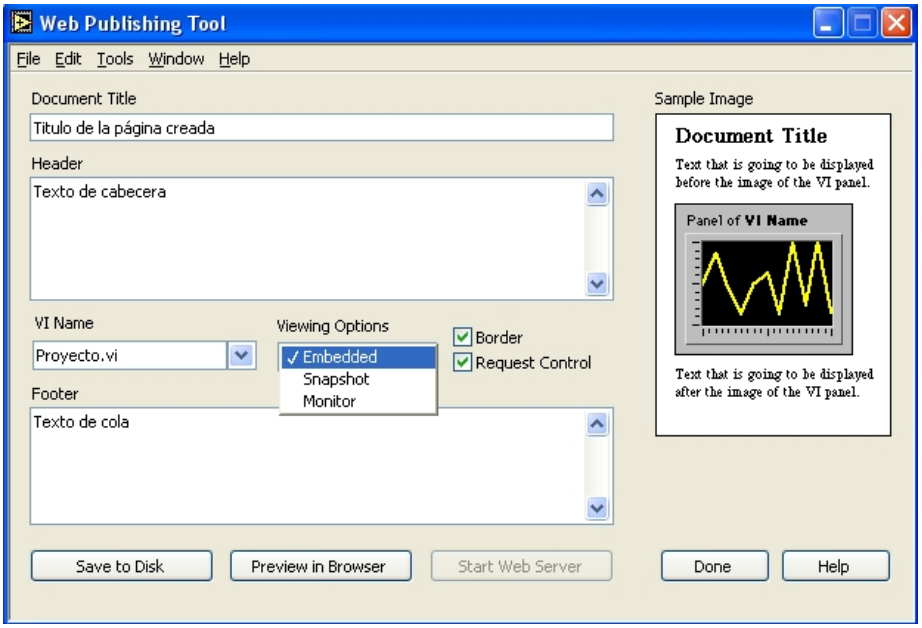


Figura 30.- Acceso al menú de configuración del editor HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">
<HTML>
<HEAD>
<TITLE>Titulo de la página ejemplo tesis</TITLE>
</HEAD>
<BODY >
<H1>Titulo de la página ejemplo tesis</H1>
Texto de cabecera<P>


<TABLE BORDER = 1 BORDERCOLOR = #000000><TR><TD>

<OBJECT ID="LabVIEWControl" CLASSID="CLSID:A40B0AD4-B50E-4E58-8A1D-8544233807AB"
WIDTH=1200 HEIGHT=730
CODEBASE="ftp://ftp.ni.com/support/LabVIEW/runtime/windows/7.0/LVRunTimeEng.exe">

<PARAM name="LVFPPVINAME" value="edsplabtesis.vi">
<PARAM name="REQCTRL" value=true>

<EMBED SRC=".LV_FrontPanelProtocol.rpvi7" LVFPPVINAME="edsplabtesis.vi"
REQCTRL=true
TYPE="application/x-LabVIEWrpvi7" WIDTH=1018 HEIGHT=708
 PLUGINSPAGE="http://www.ni.com/webappdemos/lv"></EMBED>

</OBJECT>
</TD></TR></TABLE>

<P>
Texto de cola
```

```
</BODY>
</HTML>
```

Figura 31.- Ejemplo de código generado para la página HTML

El cliente remoto no tendrá más que indicar en su navegador la dirección Web con expresión del puerto y página HTML, de tal forma que, caso de no tener instalado el plug-in “NI LabVIEW Run Time Engine”, se invita al mismo a instalarlo gratuitamente; pudiendo acceder al control del panel remoto.

El servidor Web de LabVIEW quedará a la escucha de las peticiones de los clientes que acceden al panel remoto a través de su navegador Web. Cuando el cliente realiza una petición a través del panel remoto el servidor le devuelve un código que, a través del plug-in “NI LabVIEW “Run Time Engine” es compilado y ejecutado. El mismo proceso sucede en el servidor. Por tanto, el código de la página no se compila una única vez, sino que éste está continuamente sufriendo un proceso de traducción/ejecución por parte del “Run Time Engine”.

4.5. Seguridad en el acceso al panel remoto

LabVIEW ofrece algunos mecanismos de seguridad en el acceso a los paneles remotos y el código. En primer lugar, es posible dotar al panel frontal y diagrama de bloques de una clave que impida que ningún usuario que la desconozca pueda alterar el código o apariencia empleando las opciones del propio entorno.

También ofrece la posibilidad de realizar un filtrado IP de las direcciones entrantes de los clientes, pudiendo limitar el acceso de determinadas direcciones potencialmente peligrosas o no deseadas.

El panel frontal puede ser protegido con una clave de acceso empleando ciertas funciones en el diagrama de bloques. De tal manera que se deshabiten los controles mientras no se introduzca una clave válida.

Figura 32.- Control asociado al acceso seguro

Sin embargo, el servidor Web de LabVIEW no soporta lenguajes como PHP, que nos permiten implementar sistemas de seguridad más avanzados, evitando posibles ataques al mismo.

Una posible alternativa podría ser el empleo de otro servidor distinto al ofrecido por LabVIEW, como el uso del servidor Apache, que soporta PHP y dispone de varias opciones configurables en lo que a la seguridad se refiere, siendo la solución propuesta para la gestión de la subida de los ficheros por

parte de los usuarios. Sin embargo, este tipo de servidores no soportan paneles remotos, por lo que no pueden ser empleados para este fin[11].

4.6. Discusión y conclusiones

Los distintos desarrollos del laboratorio eDSPlab, tanto en la versión únicamente instrumental [119], como en la versión que incorporaba un DSK de la familia TMS320C3x [120], y finalmente en la actual propuesta basada en un DSK de la familia TMS320C6000 de Texas Instruments [124], han puesto de manifiesto la posibilidad de acceder remotamente a un complejo sistema de instrumentación, incluido un DSP. Dicho laboratorio se ha incluido en un contexto académico con objeto de poder determinar la adecuación metodológica en su uso, [121], [122], así como la aceptación que esta herramienta tiene en la población de estudiantes [123], al mismo tiempo que se han desarrollado distintas herramientas complementarias, formas de acceso y se ha perseguido poder integrarlo en el contexto de un entorno CMS tipo Moodle, analizando distintas métricas en relación al uso técnico de la misma y percepción del usuario final [136], [137].

Se ha observado que la gestión de la seguridad en el acceso remoto en un entorno LabVIEW presenta algunas limitaciones. Resulta complejo establecer mecanismos que permitan la identificación del cliente que hace uso de los recursos; limitándose el sistema de seguridad propio del entorno al filtrado de direcciones IP y protección del panel frontal y código mediante clave. Es posible implementar un sistema de claves preestablecidas con distintos permisos, pero tenemos la imposibilidad de gestionar la creación de nuevos usuarios y claves. Además, una vez conocida la dirección Web donde se publica el panel remoto, ésta no cambia, por lo que cualquier usuario puede acceder a la misma, aunque sea en modo visualización.

El control remoto del panel es monousuario, lo que implica que no es posible establecer entornos cooperativos de trabajo donde varios usuarios operan sobre el instrumental, si bien es cierto que un único usuario puede acceder al control del mismo, el resto de clientes conectados a la aplicación podrán observar las operaciones realizadas sobre el mismo.

El uso de plataformas de pago como LabVIEW implica un coste económico considerable, el cual varía en base a las prestaciones que se precisen, diferenciándose distintas versiones. Este coste repercute en la máquina servidora únicamente en el caso de emplear el modelo de cliente delgado, basado en el plug-in “Run Time Engine” que se ejecuta sobre el navegador Web y que ofrece LabVIEW en su página Web. A lo anterior hay que sumar el coste añadido de módulos especiales para aumentar ciertas funcionalidades o en el caso de actualizaciones software.

Además, el cambio de versiones y la discontinuidad de ciertos productos ofrecidos implican que, en muchos casos, no pueda hacer el sistema ampliable

[11] National Instruments ofrece el “Internet Developers Toolkit” para LabVIEW que implementa un “G Web Server”, éste permite leer y escribir documentos XML, enviar o tomar datos de un servidor FTP, visualizar pero no controlar paneles con un navegador web a través de Internet, crear programas con interfaces CGI, no obstante, éste no soporta código PHP ni paneles remotos.

o escalable, caso del DSP Toolkit; lo que pone de manifiesto la dependencia con las políticas de la empresa proveedora, con el riesgo que ello conlleva.

La comunicación que se establece entre el usuario y el panel remoto, basada en el servidor que ofrece LabVIEW, implica un continuo proceso de ejecución-compilación para poder atender a los cambios producidos en los controles del panel. Esto repercutirá en una sobrecarga de CPU y memoria tanto en el usuario remoto como en el servidor.

El acceso remoto de los usuarios a la plataforma empleando el "LabVIEW Run Time Engine" tiene ciertas implicaciones e incompatibilidades; así, por ejemplo, la versión de Run Time Engine debe ser igual a la de la máquina servidora, por otra parte, si la máquina servidora es de 32 bits, el navegador empleado en el cliente también debe serlo.

Al estar limitado el Web Server de LabVIEW a la provisión de servicios HTTP, no pudiendo ofrecer PHP, la utilización de extensiones como la autentificación de usuarios personalizada, la gestión de subida de ficheros a ejecutar en el DSK al servidor, etc., ha supuesto la duplicación de recursos, debiendo instalarse un servidor con capacidad de manejar el lenguaje php y bases de datos SQL en paralelo al Web Server LabVIEW. Dicha duplicidad, además de suponer una sobrecarga de procesamiento, lleva ligado la necesidad de emplear dos puertos diferentes para la provisión de los servicios Web.

Lo anteriormente expuesto conlleva a que la integración con plataformas de gestión de contenidos o entornos de aprendizaje como Joomla o Moodle, entre otros, resulta una tarea tediosa y compleja; si bien no imposible, persistiendo los problemas de integración y seguridad expuestos.

Pese a las anteriores limitaciones y aspectos técnicos descritos, el laboratorio ha funcionado con normalidad atendiendo a las necesidades de los alumnos de la asignatura "Complementos de Sistemas Electrónicos Digitales", de 3er curso de Ingeniería de Telecomunicación de la Universidad de Sevilla, observándose un aceptable grado de satisfacción entre los usuarios del mismo.

5. Nueva propuesta de laboratorio de instrumentación remota: eDSPlab+

Tal y como se describe en el segundo capítulo de este trabajo de investigación, a la hora de seleccionar un determinado entorno de programación para implementar un sistema de instrumentación debemos tener en consideración varios aspectos o figuras de mérito [59]. Entre ellas cabe destacar la facilidad de uso, la flexibilidad, el rendimiento y la funcionalidad, entre otros.

El modelo de sistema de instrumentación que se propone con eDSPlab+ persigue mejorar algunas de estas figuras de mérito, tanto desde el punto de vista técnico y como por parte del usuario final [136].

eDSPlab+ desarrolla un nuevo modelo de control remoto vía Web de sistemas de instrumentación; dividido en dos bloques o capas principales: La aplicación Web, que gestionará la comunicación entre la interfaz del cliente final y el servidor y el controlador remoto, cuya función es llevar a cabo los procesos necesarios para la comunicación entre la instrumentación y el servidor.

El uso de tecnologías Web para el acceso a sistemas de instrumentación está adquiriendo un especial interés en los últimos años; encontrando incluso soluciones profesionales en otros campos como el de la automatización o la domótica donde los clásicos controladores están evolucionando hacia dispositivos conectables a redes LAN y con posibilidad de crear interfaces de monitorización y control tipo SCADA empleando como interfaz los propios navegadores Web; caso de los nuevos autómatas programables LOGO! De Siemens o el sistema inmótico basado en LonWorks Simon VITA, de la firma Simon [138],[139].

Para resolver el acceso al laboratorio de instrumentación virtual se ha dividido el problema en diferentes fases, resultando en un modelo de 5 capas, cada una de las cuales tendrá asociada una función concreta.

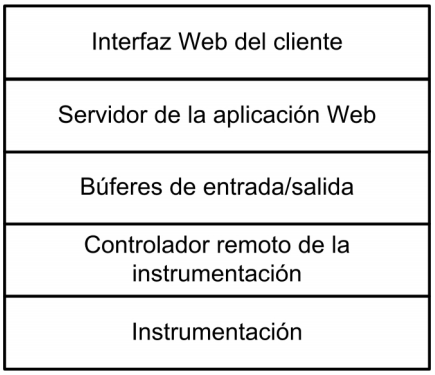


Figura 33.- Capas del modelo eDSPlab+

Además, el desarrollo íntegro de este modelo de instrumentación remota se ha diseñado empleando lenguajes y herramientas de software libre[12].

El escenario final es el de un sistema compuesto por varios instrumentos que, en el caso de la bancada de pruebas descrita anteriormente, se trata de instrumentación GPIB junto con un DSK conectados a un PC. Para que un usuario remoto pueda acceder a la instrumentación, el PC deberá actuar como servidor, presentando una interfaz Web al cliente que podrá visualizar y con la que podrá operar. Además, se deberá gestionar la comunicación con el sistema de instrumentación, tanto GPIB como el DSK. Finalmente, se deben desarrollar los mecanismos necesarios para intercomunicar estos dos bloques, de tal manera que las peticiones de un cliente remoto conectado a la interfaz Web sean enviadas al instrumento remoto y las respuestas del mismo, sean reflejadas en la interfaz Web del cliente.

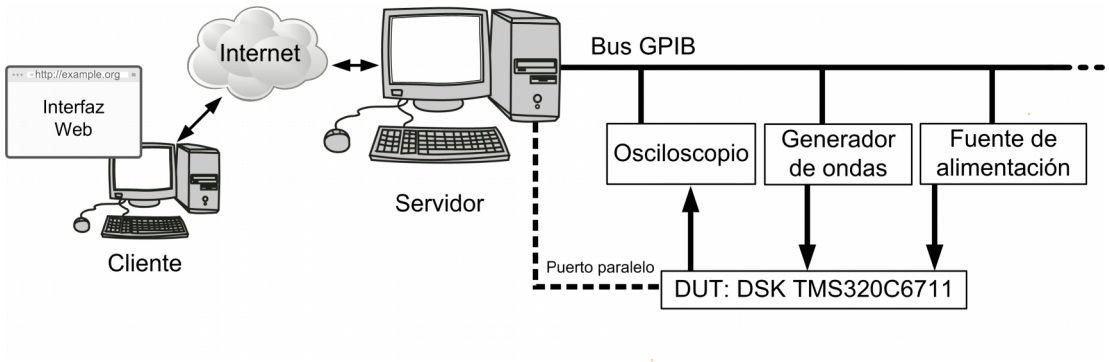


Figura 34.- Escenario final del laboratorio de instrumentación remota eDSPlab+

5.1. Elección del lenguaje y plataformas de programación

Una de las principales motivaciones del presente trabajo de investigación ha sido reducir la dependencia en el desarrollo de modelos de instrumentación remota con aplicaciones o entornos de desarrollos comerciales que impliquen un desembolso económico que, en muchas ocasiones no está al alcance de organismos, empresas o instituciones que desean implementar arquitecturas de instrumentación remota. El uso del software libre, pese a que puede presentar algunos inconvenientes, manifiesta un gran número de ventajas, entre las que destacan un reducido o nulo coste de utilización, el aumento en la flexibilidad de programación por parte del desarrollador y la ausencia de restrictivas y costosas licencias asociadas a las funcionalidades más elevadas.

Para dar solución a las distintas capas que constituyen eDSPlab+ se ha decidido emplear un conjunto de herramientas gratuitas basadas en JAVA[13]. Existen distintos entornos de desarrollo gratuitos para este lenguaje, lo que implica un coste cero. Se trata de un lenguaje multiplataforma y portable, independizando el sistema operativo y la arquitectura hardware de la aplicación. Además, dispone de varios elementos que serán fundamentales

[12] A excepción del entorno Code Composer Studio de Texas Instruments, necesario para implementar las funcionalidades de manejo remoto de la consola y, por tanto, la comunicación con el mismo.
[13] Si bien es cierto que será preciso hacer uso del lenguaje C en una parte del modelo, para poder acceder a las librerías que operan sobre la instrumentación GPIB y comunicarse con JAVA a través de JNI.

para dar solución a las cuestiones que se deben resolver, como el acceso a la instrumentación, lo cual se ha conseguido gracias al uso de la API de JAVA JNI (Java Native Interface).

Uno de los principales inconvenientes de Java es que, al tratarse de un lenguaje compilado e interpretado, empleándose un intérprete de Java según el sistema operativo y máquina; éste tiene un tiempo de ejecución más lento que otros lenguajes.

5.2. Acceso a la instrumentación a través de la interfaz nativa JNI

Uno de los primeros aspectos que debemos resolver en un sistema de instrumentación es cómo accede el servidor a dichos instrumentos.

Los distintos instrumentos controlables remotamente, tales como osciloscopios, fuente de señal, analizadores de espectro, etc., suelen ir acompañados de librerías de funciones en forma de drivers o DLL[14] que son utilizadas por los entornos de alto nivel para comunicarse con los mismos. Tal es el caso de la instrumentación GPIB. Sin embargo el lenguaje JAVA no puede emplear directamente una DLL.

Para resolver el problema de comunicación de la aplicación principal del servidor, basada en lenguaje JAVA, con los drivers de los instrumentos, se hace uso del Java Native Interface (JNI) [140] , un framework de programación que posibilita que los lenguajes escritos en lenguaje Java puedan interactuar con programas escritos en otros lenguajes como C, C++ o el lenguaje ensamblador, entre otros.

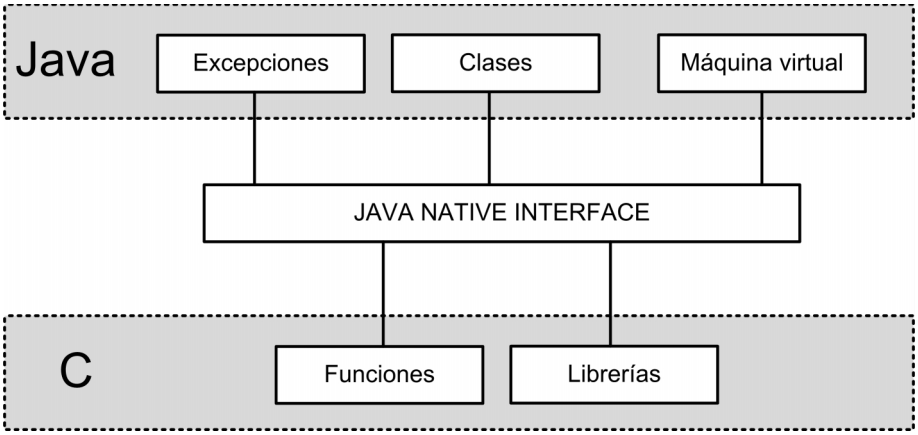


Figura 35.- Comunicación entre Java y C gracias a JNI

Gracias al empleo de la API de Java JNI, es posible utilizar código nativo, es decir, código escrito en un lenguaje distinto al lenguaje Java e incluirlo en una aplicación como si se tratara de código Java. La aplicación Java del servidor podrá, por tanto, comunicarse con una aplicación escrita en lenguaje C y ésta, a su vez, con una DLL, resolviendo así el problema de comunicación con el sistema de instrumentación [141].

[14] DLL es una denominación exclusiva a los sistemas operativos Windows, en el caso de sistemas operativos linux serían archivos “so”.

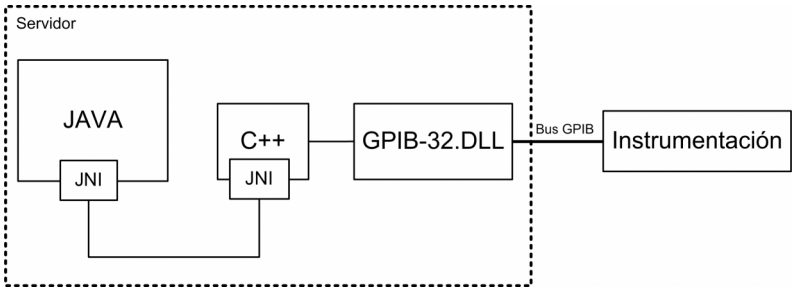


Figura 36.- Esquema de acceso al sistema de instrumentación empleando JNI

Entre los archivos que incluye el software de la tarjeta GPIB instalada en el PC y que actúa como maestro de la comunicación con los equipos de instrumentación, emplearemos los ficheros ni488.h y GPIB-32.dll. El fichero ni488.h será el que nos permita acceder mediante punteros a las funciones definidas en el estándar GPIB. Por otra parte, la definición de dichas funciones vendrá recogida en la librería dinámica GPIB-32.dll.

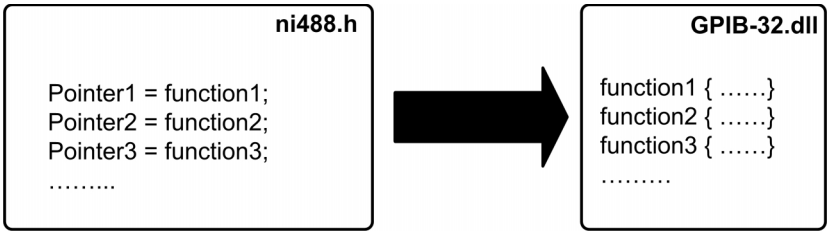


Figura 37.- Relación entre el driver GPIB-32.dll y el archivo ni488.h

Las funciones GPIB que será preciso emplear para el control y manejo de la instrumentación son las que se muestran en la siguiente tabla:

Nombre	Relación de parámetros	Funcionalidad
(int) ibdev	(int BdIndx, int pad, int sad, int tmo, int eot, int eos) BdIndx Indica la interfaz de acceso para el dispositivo. pad Dirección primaria GPIB del dispositivo. sad Dirección secundaria del instrumento. tmo Valor del temporizador de entrada/salida. eot modo EOI del dispositivo. eos carácter EOS y modos.	Esta función obtiene el identificador del instrumento.
(int) ibrd	(int ud, void *rdbuf, long count) ud Descriptor de dispositivo. count Número de bytes a ser leídos del bus GPIB.	Se encarga de leer cadena de caracteres por el bus GPIB y devuelve los errores producidos en la lectura a través de la variable ibsta.
(int) ibwrt	(int ud, void *wrtbuf, long count) ud Descriptor del dispositivo. wrtbuf Dirección del buffer que contiene los bytes del comando a escribir. count Número de bytes a ser escritos.	Su función es la de escribir una cadena de caracteres por el bus GPIB y devolver los errores producidos en la escritura a través de la variable ibsta.
(int) ibonl	(int ud, int v)	Tiene por objeto liberar el identificador del

	ud descriptor de dispositivo o interfaz. v Indica que interfaz o dispositivo se pone offline.	instrumento poniéndolo fuera de línea. También devuelve el estado del instrumento a través de la variable ibsta.
(int) ibclr	(int ud) ud descriptor de dispositivo o interfaz.	Reinicializa el instrumento especificado en ud y devuelve el estado del instrumento a través de la variable ibsta.

Tabla 3.- Listado de funciones empleadas de la librería dinámica GPIB-32.dll

De tal forma que para manejar un instrumento a través del bus GPIB empleando dichas funciones se tendrá, en primer lugar, que inicializar el instrumento empleando la función *ibdev*, seguidamente se leerán o escribirán los comandos GPIB empleando las funciones *irbd* y *ibwrt*, respectivamente. Por último, cuando finalice el control del instrumento se pondrá fuera de línea empleando la función *ibonl*.

Un aspecto importante que debemos tener en cuenta en el diseño del modelo del sistema de instrumentación basado en GPIB es que el estándar SCPI no especifica exactamente el lenguaje de los comandos GPIB, sino un estilo de formato; en otras palabras, los comandos SCPI no tienen porqué ser idénticos en todos los instrumentos, únicamente comparten unas normas o patrones [31]. Para evitar la dependencia del sistema con los equipos de instrumentación, los comandos asociados a cada equipo son almacenados en un archivo de texto, de tal forma que cada equipo tendrá su propio archivo de comandos específicos y se accederá a ellos a través de la clase *Properties* de la API de Java, que nos permite crear ficheros de configuración complejos y manejarlos de forma sencilla. Con esto se consigue aumentar la independencia con el equipo de equipo de instrumentación.

Por tanto, para controlar las funciones que gestionan el bus GPIB se crea una librería dinámica “mygpib.dll”. Dicha librería dinámica carga la librería GPIB-32.dll e implementa dos métodos genéricos, “Write” y “WriteRead”. La función “Write” impondrá el comando GPIB que se le pasa como parámetro[15]. Por otra parte, la función “WriteRead” impone el comando GPIB que se le pasa como parámetro y lee del mismo una respuesta que se le pasa como parámetro también.

[15] Comando que estará especificado en el archivo “properties” del instrumento en cuestión en base a la acción del usuario remoto sobre la interfaz gráfica.

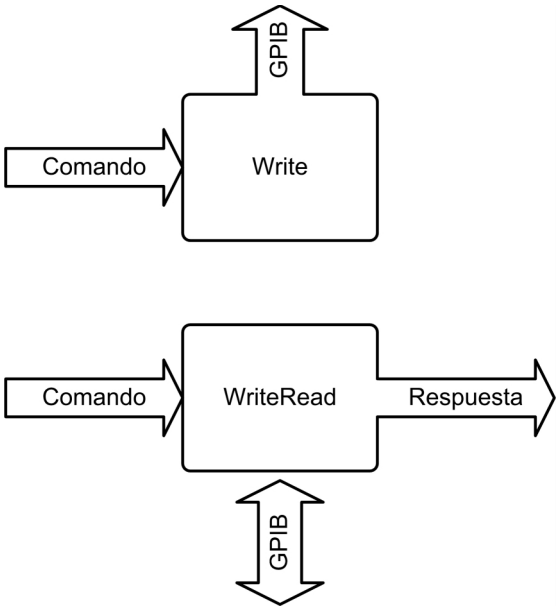


Figura 38.- Funciones “Write” y “WriteRead” de la DLL “mygpib.dll”

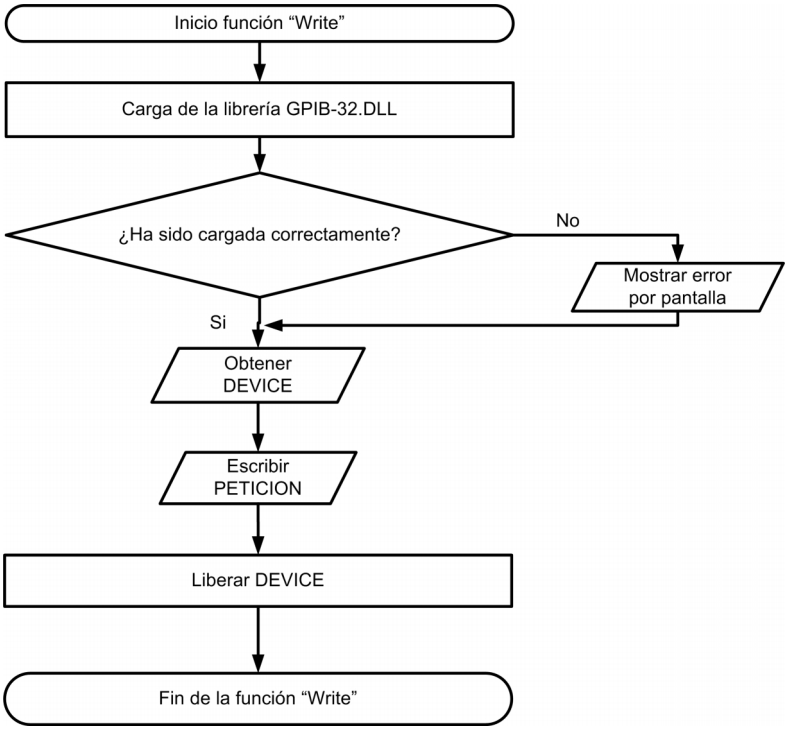


Figura 39.- Diagrama de flujo de la función Escribir

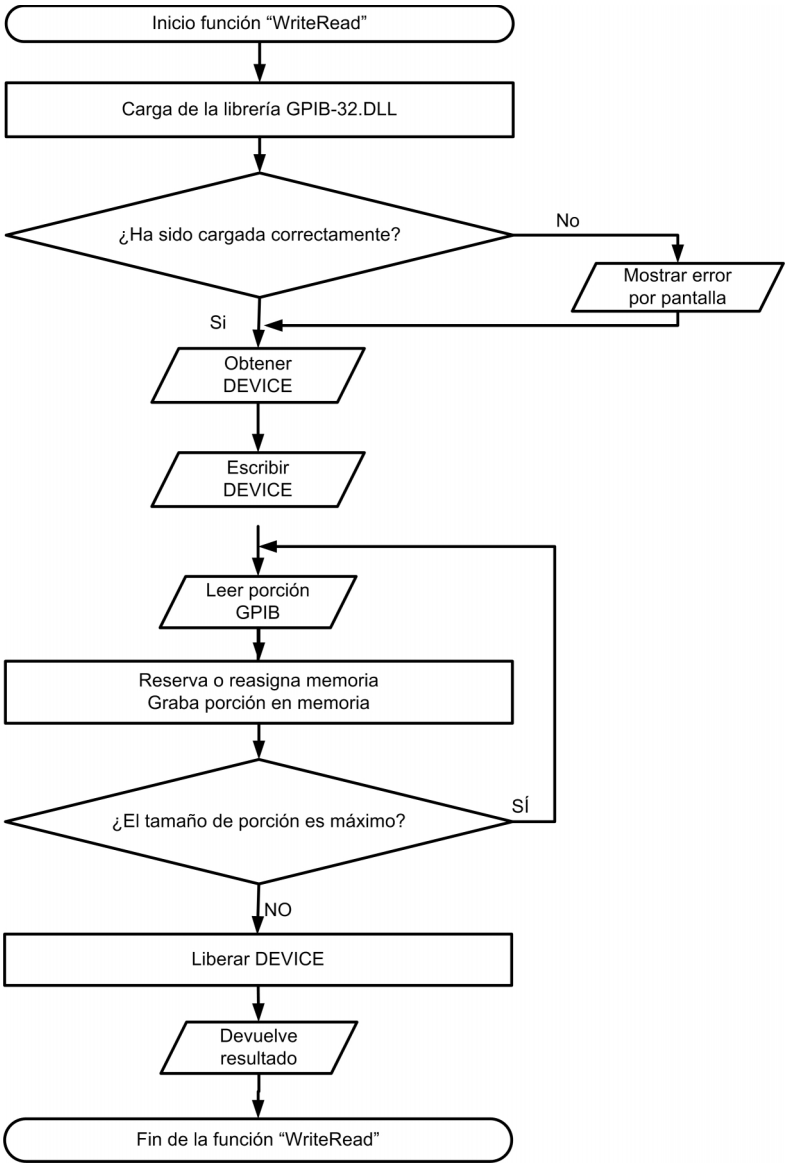


Figura 40.- Diagrama de flujo de la función EscribirLeer

Las funciones escritas en lenguaje C “Write” y “WriteRead” serán compiladas para crear la librería dinámica, “mygpib.dll”, que permita acceder a las mismas desde la aplicación JAVA empleando JNI, incluyendo el fichero de cabecera (.h) que contiene las declaraciones de las funciones nativas que serán empleadas desde JAVA.

El fichero que contiene las declaraciones nativas se generará al crear previamente una clase en Java que contendrá las declaraciones de las funciones “Write” y “WriteRead” escritas en lenguaje nativo y que será la que nos permita acceder a ellas desde el lenguaje Java.

La clase “ControlGPIB.java” incluirá la declaración de las dos funciones “Write” y “WriteRead” escritas en lenguaje nativo C y el nombre de la librería dinámica desde donde será cargado el contenido de estas funciones.

Para la identificación de los distintos instrumentos, el envío de comandos y el manejo de los ficheros de comandos asociados a los distintos equipos, se

han creado 3 nuevas clases: La clase “Instrumento”, la clase “Comando” y la clase “Instrucción”.

La clase “Instrumento” contendrá las características que definen a cada instrumento, como son la interfaz, la dirección principal y secundaria, un identificador, un índice y un estado, definiendo dos métodos cuya función será la de rellenar el identificador del instrumento y comprobar el estado del mismo. Existirán tantos objetos “Instrumento” como equipos de instrumentación haya físicamente.

Una vez creado un objeto de tipo “Instrumento” se realizará la carga del archivo *properties* correspondiente a dicho instrumento empleando la clase “Instrucciones”, que implementa un único método encargado de dicho proceso, el método “Carga”.

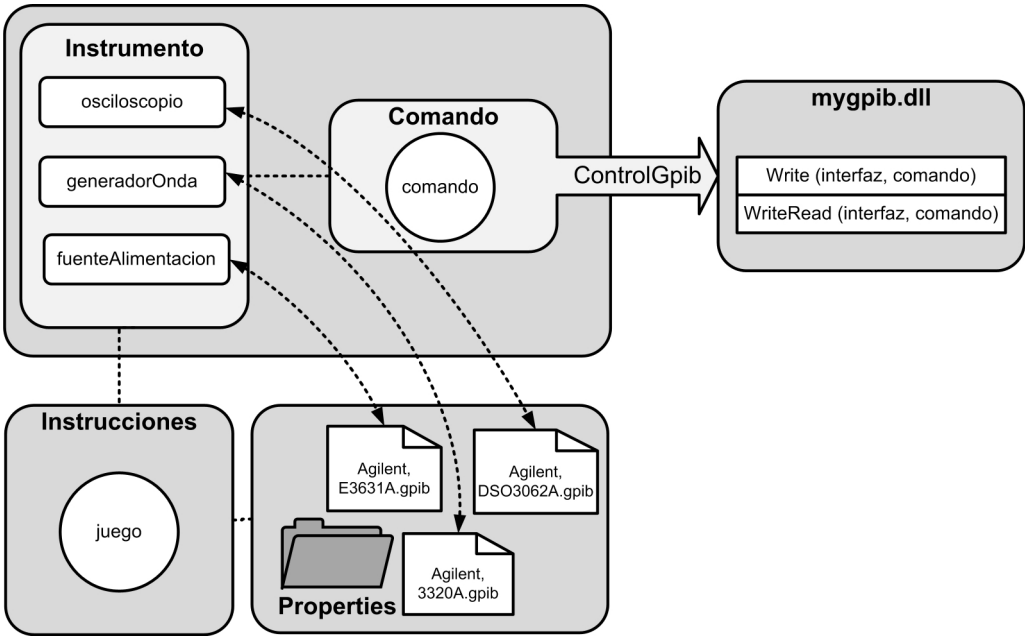


Figura 41.- Acceso a los archivos “properties” y flujo de información

La utilización de los archivos “properties” permitirá aislar las dependencias que existen en los comandos de los distintos equipos del mismo tipo, de forma que puedan ser reemplazables. El usuario remoto indicará a la aplicación principal del servidor una “clave genérica de comando” seleccionado y, a través del fichero “properties” del instrumento asociado, se hará la correspondencia con el comando GPIB real, que será el que se envíe a la interfaz a través de la clase “Comando”.

```
# COMMON COMMANDS

idn = *IDN?
rst = *RST
cls = *CLS
opc = *OPC?

# ROOT COMMANDS

auto = :AUTO
run = :RUN
stop = :STOP

# CHANNEL COMMANDS

canal1on = :CHAN1:DISP1
canal2on = :CHAN2:DISP1
canal1off = :CHAN1:DISP0
canal2off = :CHAN2:DISP0
canal1disp = :CHAN1:DISP?
canal2disp = :CHAN2:DISP?

# WAVEFORM DATA

onda = :wav:data?
```

Figura 42.- Vista de un archivo tipo "properties"

Por otra parte, la clase "Comando" implementará un único método, "mandar", que enviará el comando especificado[16] al instrumento que haya sido creado con antelación. Dicho método hará uso de las definiciones realizadas en el fichero de cabecera "ControlGPIB.java" de las funciones programadas en C para la comunicación con el bus GPIB a través de JNI.

5.3. Comunicación con la interfaz gráfica del cliente remoto

Tal y como hemos comentado, la aplicación del servidor de eDSPlab+ se divide en dos bloques principales. Uno de ellos se encarga de la gestión de la comunicación con los instrumentos de medida y la tarjeta del DSK. Por otra parte, el segundo bloque será el responsable de presentar un flujo de información entre la interfaz Web del usuario remoto y el sistema local.

Para facilitar la comunicación entre el servidor y el cliente remoto a través de una interfaz Web se crearán dos ficheros que actuarán como búferes o colas de entrada-salida; de esta forma se elimina la dependencia entre las tecnologías empleadas para diseñar el entorno gráfico del cliente para acceder al sistema y el propio servidor Java local.

La aplicación principal del servidor se encontrará a la espera de los comandos recibidos por la interfaz Web de los clientes remotos. Cuando se reciba un comando por parte del cliente, el programa principal invocará a los métodos de los objetos asociados a las diferentes clases descritas en el anterior apartado, que gestionarán la comunicación con los instrumentos

[16] El cual vendrá impuesto a través de la interfaz del usuario remoto y cuya sintaxis dependerá del archivo "properties" del mismo.

reales; traduciéndose en comandos GPIB. Cuando dichos comandos precisen de respuesta, la información devuelta por el bus será puesta a disposición de la interfaz Web del cliente remoto.

El método de intercambio de información entre la interfaz Web del cliente remoto y el servidor local de control de la instrumentación se basa en la implementación de dos colas o búferes FIFO (First-In First Out), uno de entrada y otro de salida. En dichas colas, implementadas mediante el empleo de sendos ficheros de texto, los comandos procedentes de la interacción con un instrumento remoto de la interfaz Web se volcarán en la cola de entrada. Por otra parte, la respuesta a las peticiones realizadas a los equipos será volcada en el búfer de salida. La sintaxis de comunicación deberá ser conocida por ambas partes.

La principal ventaja del empleo de colas FIFO en el proceso de interacción entre la interfaz Web y el servidor local de instrumentación se basa en la total independencia entre ambos, pudiendo emplearse cualquier lenguaje o plataforma para crear las interfaces cliente, sin más que respetar la sintaxis del protocolo establecido.

La aplicación principal del servidor, por tanto, creará los instrumentos que se estén empleando, cargará el juego de instrucciones asociado, y quedará leyendo el búfer de entrada a la espera de alguna petición; cuando se recibe, traduce el comando genérico recibido de la interfaz del cliente en un comando específico del instrumento; gracias a los ficheros “properties” de cada uno de ellos, comunicándose con la instrumentación y, en caso de ser preciso, recibirá la respuesta que volcará en el búfer de salida.

En particular, se ha implementado este modelo sobre los dispositvos que componen la bancada de ensayos descrita en el documento, por lo que el sistema final tiene la siguiente estructura.

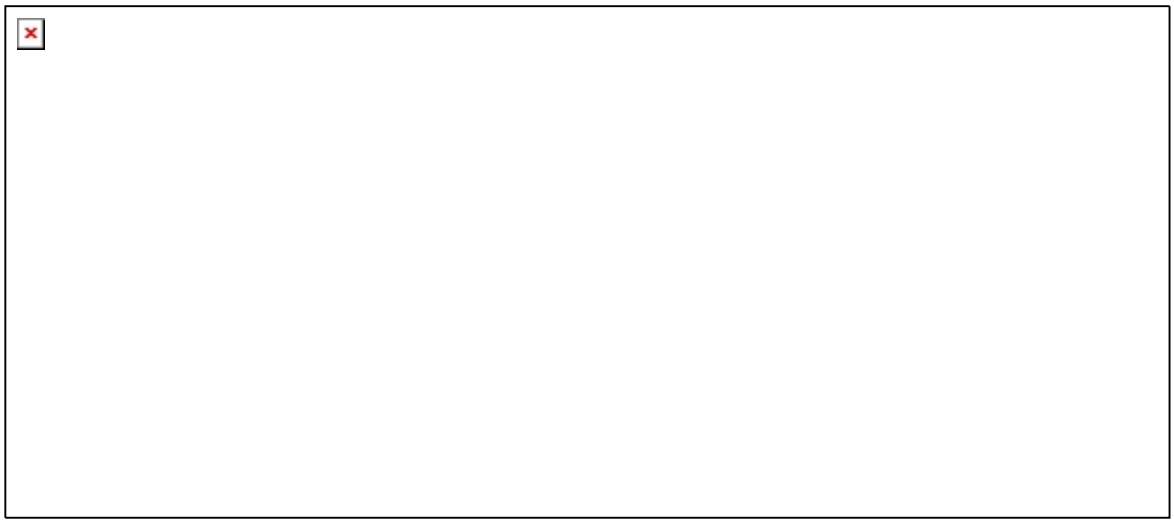


Figura 43.- Modelo de comunicación basado en la utilización de búferes

5.4. Interacción con el DSK empleado el lenguaje Perl

Otro de los elementos que forma parte de nuestra bancada de instrumentación es el sistema de desarrollo (DSK) basado en el DSP

TMS320C6711, tal y como se ha comentado anteriormente. Para resolver la comunicación del DSK con LabVIEW, éste dispone del "DSP Test Integration Toolkit[17]", el cual ofrece un conjunto de funcionalidades que facilitan realizar ciertas operaciones sobre el mismo a través del programa Code Composer Studio.

Para comunicarnos con el DSK trataremos de seguir una estructura similar al modelo propuesto con el sistema de instrumentación; no obstante, para simplificar la solución adoptada se ha decidido hacer uso de la herramienta Ccs_scripting que el entorno Code Composer Studio incorpora desde su versión 2.1 [142], la cual permite interactuar con el entorno de desarrollo del DSP sin manipular el programa físicamente, lo cual es posible a través de programas escritos en lenguaje Perl, es decir, ficheros Perl [143].

Para ejecutar el archivo Perl que permite interactuar con las funcionalidades de Code Composer Studio se precisa hacerlo desde la consola de comandos.

Para que nuestra aplicación principal, basada en lenguaje Java, pueda acceder a las funcionalidades que el entorno Code Composer Studio ofrece, dicha aplicación Java llamará a un script que se ejecutará bajo MS-DOS el cual, a su vez invocará al archivo Perl asociado que se comunicará, empleado la herramienta Ccs_Scripting con el DSK vía Code Composer Studio.

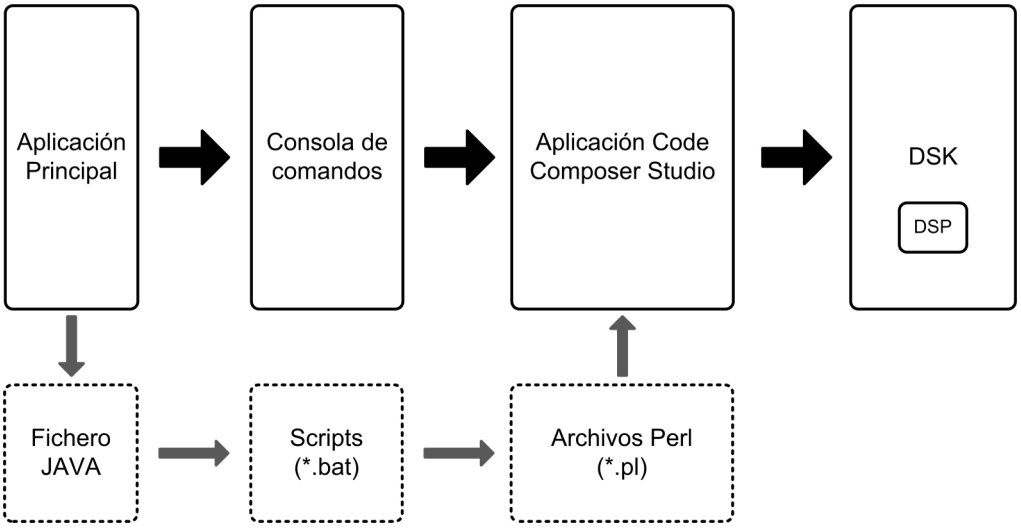


Figura 44.- Modelo de invocación del Ccs_scripting desde Java

Para la comunicación con el usuario remoto se emplea un archivo de texto, "INDSP.txt", que actúa a modo de cola de servicio FIFO, donde se volcarán las peticiones remotas del cliente que serán procesadas por la aplicación Java responsable de la comunicación con el DSK; atendiendo las peticiones por orden de llegada.

Por tanto, para la ejecución de cada acción por parte del usuario remoto[18], existirán dos ficheros asociados; un fichero perl, cuya ejecución

[17] En la actualidad este Toolkit se encuentra integrado en el entorno LabVIEW.
[18] Se han implementado las acciones "Load", que permite la carga de un proyecto en la memoria del DSP, "Build", compila el proyecto convirtiéndolo en un archivo ejecutable ".out", "Run", ejecuta el archivo

materializa la acción deseada sobre el programa Code Composer Studio y un archivo .bat, necesario para establecer la llamada al archivo perl desde el programa Java.

```
cd ..
cd ..
cd ..
cd ..
cd CCStudio_v3.3/bin/utilities/ccs_scripting
set
path=C:\Perl\bin;C:\CCStudio_v3.3\bin\utilities\ccs_scripting;%path%
set PERL5LIB=C:\Perl\bin;C:\CCStudio_v3.3\bin\utilities\ccs_scripting
cd examples/Perl
perl ccs_run.pl
```

Figura 45.- Ejemplo del fichero .bat asociado a la ejecución de programa en DSP

```
use CCS_SCRIPTING_PERL;
my $MyCCScripting = new CCS_SCRIPTING_PERL::CCS_Scripting();
my $MyCCScripting2 = new CCS_SCRIPTING_PERL::CCS_Scripting();
my $sCurDir;
my $sLogFile;
my $sProject;
my $sProgram;
my $nPCVal;
my $nPCVal2;
my $nMainAddr;
my $nBreakpoint1;
my $nBreakpoint2;
my $sHexOutput;
my $sBoardName;
my $sCPUName;
my $sVersion;

$sProgram = ".\\ejemplo_tesis.out";
$sLogFile = ".\\ccs_run.log";
$sCurDir = ".\\";

print ("Testing...\n");

$MyCCScripting -> ScriptTraceBegin($sLogFile);
$MyCCScripting ->
ScriptTraceVerbose($CCS_SCRIPTING_PERL::VERBOSE_ALL);

$sVersion = $MyCCScripting->ScriptGetVersion();
$MyCCScripting -> ScriptTraceWrite("$sVersion\n");
print "$sVersion\n";
$MyCCScripting -> CCSOpen($CCS_SCRIPTING_PERL::ISA_C67,
                    0,
                          0,
                          $CCS_SCRIPTING_PERL::PLATFORM_EMULATOR,
                    1);

$MyCCScripting -> TargetReset;
```

.out previamente volcado en memoria, “Halt”, que detiene la ejecución del archivo .out que está ejecutándose y “Open”, que permite abrir un fichero “.out”.

```
$MyCCScripting -> TargetRestart;
$MyCCScripting -> TargetRun;
```

Figura 46.- Ejemplo de fichero Perl asociado a la ejecución de programa en el DSP

Un aspecto a destacar es que, algunas acciones, como son la carga, apertura o compilación de un proyecto, requieren que se indique la ruta del archivo donde el proyecto está localizado, mientras que otros comandos, como la ejecución o la detención de la ejecución no. Esto implica la necesidad de modificar la ruta y nombre del proyecto de cada cliente remoto en los archvos Perl que posteriormente serán invocados.

La aplicación principal Java extraerá del búfer de entrada, “Indsp.txt”, la información del comando que el cliente remoto desea ejecutar y la ubicación y nombre del proyecto. En el caso de que el comando a ejecutar precise de dicha ruta, se modificará el archivo Perl asociado a dicha acción, actualizando esta información, en caso de tratarse de comandos que no precisan de dicha información no se actualizaría el fichero Perl asociado. Seguidamente se llamaría desde la aplicación al archivo .bat asociado que que invoque al archivo Perl que gestiona la comunicación con el Code Composer.

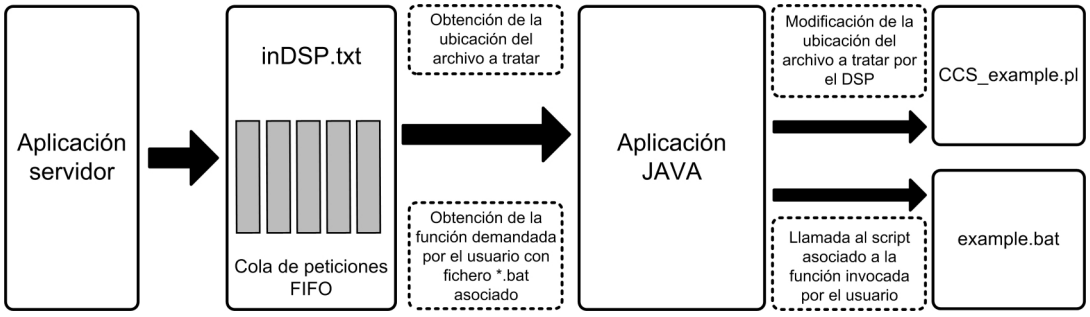


Figura 47.- Esquema de funcionamiento

5.5. Interfaz Web

Para la comunicación del cliente remoto del sistema de instrumentación con el controlador local se propone el diseño de un modelo basado en “Aplicaciones Web”. Las aplicaciones Web son herramientas que permiten la utilización de un navegador para que los usuarios accedan a los recursos de un servidor remoto a través de una red, normalmente, Internet. Es decir, son aplicaciones codificadas en lenguajes de programación soportados por los navegadores, a los que se les “confían” la ejecución de los mismos [144].

Las aplicaciones Web son clientes ligeros que únicamente precisan de un navegador para su ejecución, lo que implica una total independencia con el sistema operativo; reduciendo los costes de actualización y mantenimiento de las aplicaciones, al no requerirse la instalación de ningún software en el lado del cliente; al mismo tiempo que resultan un entorno sencillo e intuido al usuario final [144].

Las tecnologías Web estándar, como el lenguaje HTML, presentan algunas limitaciones en las funcionalidades que ofrecen; especialmente en

cuanto a la interactividad con los elementos de la interfaz y el intercambio de información con el servidor remoto.

Existen lenguajes que permiten establecer mecanismos para que tenga lugar el intercambio de información entre el cliente-servidor, recargando la página que se muestra al cliente por cada petición, lo que posibilita mostrar la nueva información solicitada.

Para aumentar la interactividad en el extremo del cliente sin tener que hacer uso constantemente de peticiones al extremo servidor[19], se emplean distintos lenguajes Script, tales como JavaScript.

En la actualidad se están desarrollando tecnologías que permitan coordinar estos lenguajes del lado del cliente con la tecnología empleada en el extremo del servidor, tal es el caso de AJAX [146], [147], que emplea una combinación de varias tecnologías con tal propósito.

El principio de funcionamiento propuesto que permite interactuar al cliente con el sistema de instrumentación se basa en el uso de dos tecnologías: AJAX, en el lado del cliente y Servlets, en el extremo del servidor.

5.5.1. Servlets versus CGI en el extremo servidor

Un Servlet es una clase del lenguaje Java que permite ampliar las capacidades de un servidor. Si bien es cierto que los Servlets pueden atender a cualquier tipo de peticiones, éstos se emplean comúnmente para extender las aplicaciones que se encuentran alojadas en los servidores Web, siendo la contrapartida Java de otras tecnologías de contenido dinámico Web como PHP o ASP.Net, entre otras.

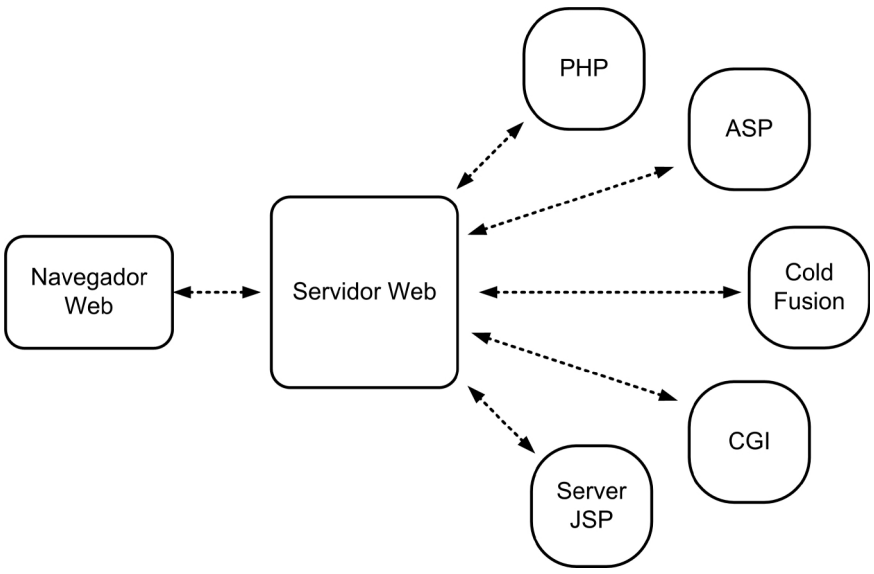


Figura 48.- Posibles tecnologías de servicios Web

Los Servlets admiten peticiones a través del protocolo HTTP desde un navegador, las procesan y devuelven una respuesta al navegador, generando

[19] Sin necesidad de recargar la página, lo que implica cierto tiempo y tráfico de red y, por tanto, molestias para el cliente remoto.

páginas Web dinámicas a partir de los parámetros de la petición recibida por el navegador del cliente Web [144], [145].

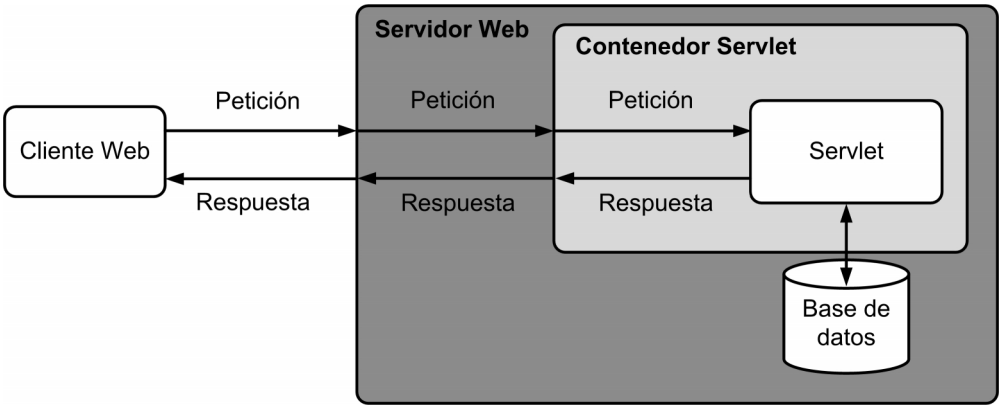


Figura 49.- Proceso de petición cliente Web-Servlet

Los formularios HTML son un mecanismo que permite revertir el sentido habitual de envío de información cliente-servidor. Gracias al empleo de los formularios HTML, el cliente Web puede hacer llegar información al servidor a través de los campos del mismo, tales como selectores, campos de texto, botones, etc. Dicha información será remitida al servidor cuando se ejecuta la acción "submit" del mismo. A su vez, los formularios tienen asociados el campo "Action" que indica la dirección a la que se enviará la información; habitualmente asociada a un correo electrónico o una aplicación, normalmente un programa CGI (Common Gateway Interface)[20], es decir, el nombre del programa del servidor encargado de procesar la información de dicho formulario.

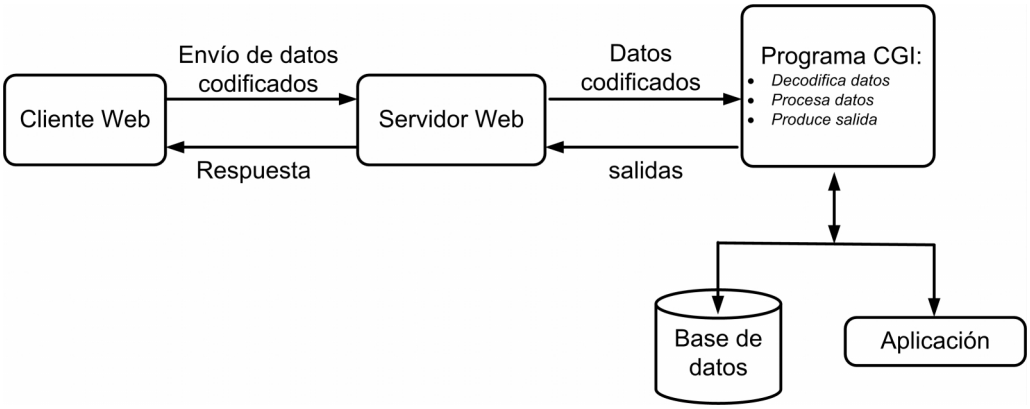


Figura 50.- Proceso de comunicación en aplicaciones servidor con CGI

Al recibir la petición del formulario del cliente, el servidor ejecutará la aplicación asociada, procesando la información contenida, interactuando con una base de datos si es preciso y enviará al cliente una página HTML de respuesta.

[20] Un ejemplo de empleo del campo "Action" podría ser:
<FORM METHOD=GET ACTION="http://www.edsplabplu.net/cgi-bin/miscript.cgi">

Para el proceso de intercambio de información entre el servidor y el cliente Web se han empleado tradicionalmente los formularios en conjunción con los programas CGI, que pueden estar escritos en cualquier lenguaje de programación, como Perl o C++, entre otros.

Una alternativa a la utilización de los programas CGI es el uso de los Servlets de Java, que nos ofrece las mismas ventajas que el propio lenguaje Java; pues no es más que una clase del lenguaje, heredando todas sus propiedades. Entre las virtudes del uso de los Servlets podemos destacar su portabilidad, en contrapartida a los programas CGI que deben ser compilados para el sistema operativo del servidor en el que se ejecutan [145].

Otro aspecto fundamental es lo que afecta al rendimiento, pues mientras los programas CGI precisan ser cargados tantas veces como peticiones de servicio tengan lugar por parte de los clientes, los Servlets, una vez son invocados, quedan activos en la memoria del servidor hasta que la aplicación del servidor los desactiva, minimizando en la mayoría de los casos el tiempo de respuesta del mismo. Cuando se dispone de múltiples clientes conectados al servidor, la aplicación CGI puede ser llamada por varios clientes simultáneamente, lo que puede provocar la saturación del mismo [145].

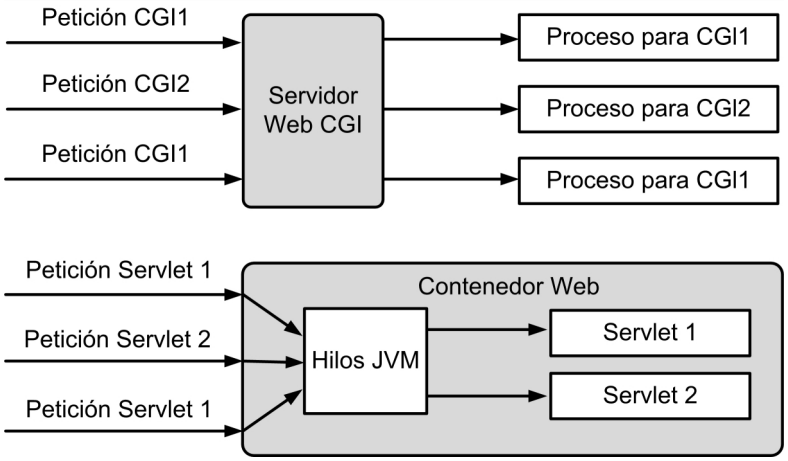


Figura 51.- Comparativa de ejecución de procesos con CGI y Servlets

Sumado a la portabilidad que ofrece el uso de los Servlets, siendo independientes del servidor y sistema operativo alojado, otras virtudes del uso de los Servlets son que permiten capturar valiosa información del cliente, como su dirección IP, de forma sencilla, tienen la posibilidad de emplear cookies y sesiones; lo que abre la posibilidad de personalizar la interacción cliente-servidor al poder guardar información específica del mismo y que pueden actuar como interfaz cliente-servidor en arquitecturas de 3 capas, ofreciendo un enlace entre el cliente y distintas bases de datos u otras aplicaciones[21].

[21] En el modelo propuesto, la ejecución de los Servlets actuarán como enlace entre la interfaz cliente y el controlador de instrumentación.

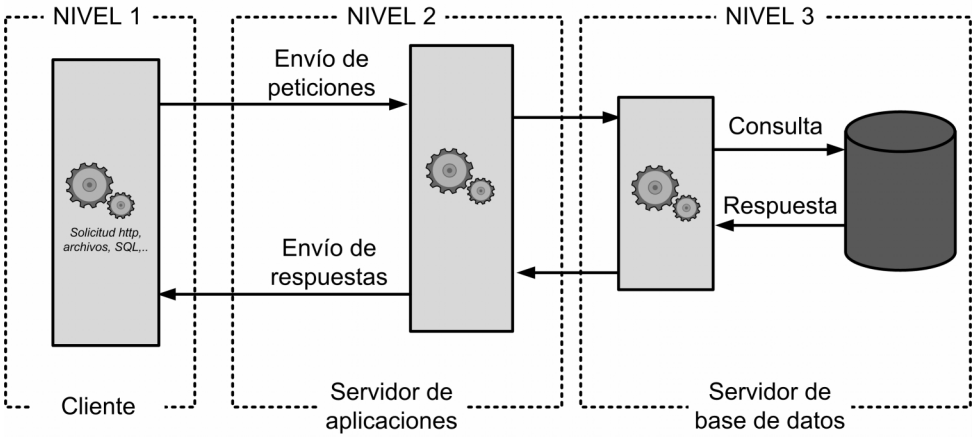


Figura 52.- Modelo de arquitectura cliente-servidor de tres capas

5.5.2. La tecnología AJAX en el extremo cliente

El modelo clásico de interacción entre un cliente y un servidor en las aplicaciones Web se basa en el uso de formularios que, al ser enviados, son procesados por el servidor y, en base a la información contenida en el formulario recibido, construye una nueva página Web, elemento a elemento, incluyendo etiquetas y contenido, que devuelve al cliente.

Sin embargo, este proceso es poco eficiente, gran parte de la página HTML enviada como respuesta a la petición del usuario ya estaba presente la primera página Web; desperdiciándose mucho ancho de banda y tiempo de procesado, lo que limita el grado de interactividad que se puede conseguir con este tipo de clientes Web, al existir elevados tiempos de espera entre las peticiones del cliente y la respuesta del servidor.

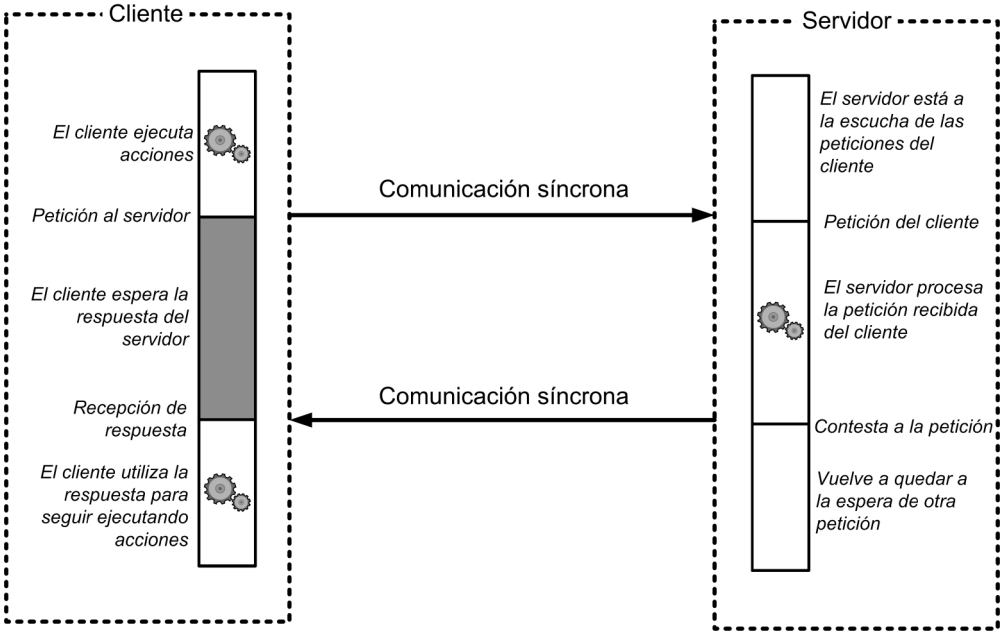


Figura 53.- Cronograma del modelo cliente-servidor síncrono

A dicho modelo se le conoce como modelo síncrono cliente-servidor, en contraposición con el modelo cliente-servidor asíncrono; que es el que se

propone con el uso de la tecnología AJAX (Asynchronous JavaScript And XML) [147].

El empleo de la tecnología AJAX permite al cliente Web enviar peticiones al servidor Web y recibir del servidor, exclusivamente, la información que se precise, empleando SOAP[22] o algún lenguaje de servicios Web basado en XML. Para procesar la respuesta del servidor, el cliente Web empleará el lenguaje JavaScript.

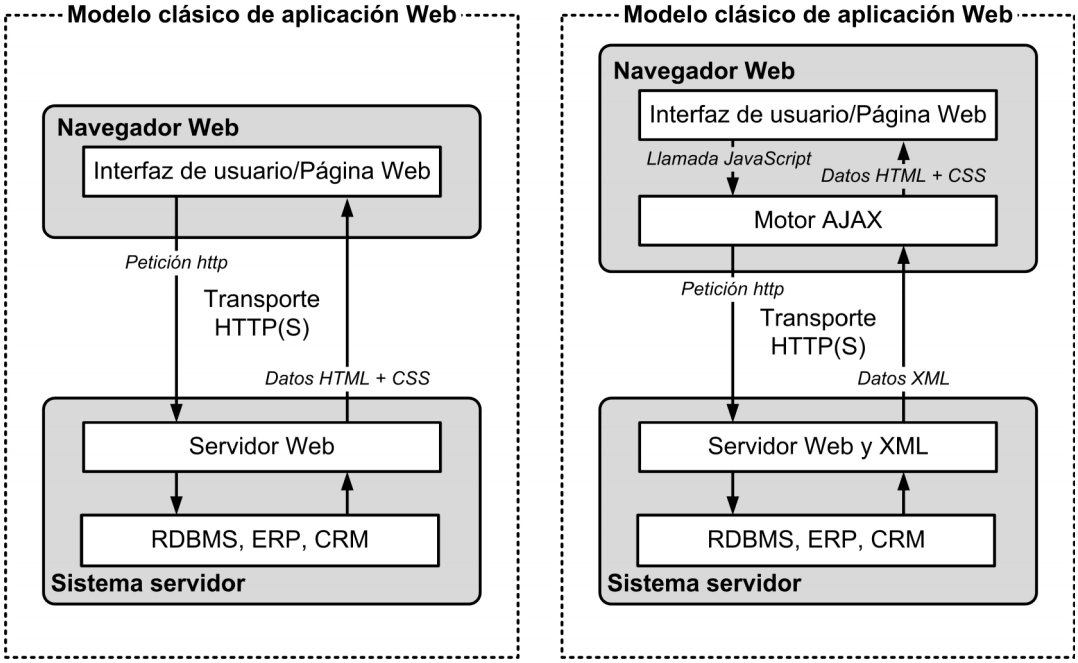


Figura 54.- Izquierda.- Modelo clásico de aplicación Web. Derecha: Modelo de aplicación Web basado en AJAX

El diseño de interfaces Web basadas en AJAX nos permitirá eliminar la necesidad de recarga constante de las páginas Web frente a peticiones de información por parte del cliente gracias a la aparición de una "capa intermedia" en la comunicación entre cliente final y el servidor, que pasa a ser asíncrona.

[22] SOAP (Simple Object Access Protocol) es un protocolo estándar que define la manera en que dos objetos en distintos procesos pueden comunicarse gracias al intercambio de datos XML.

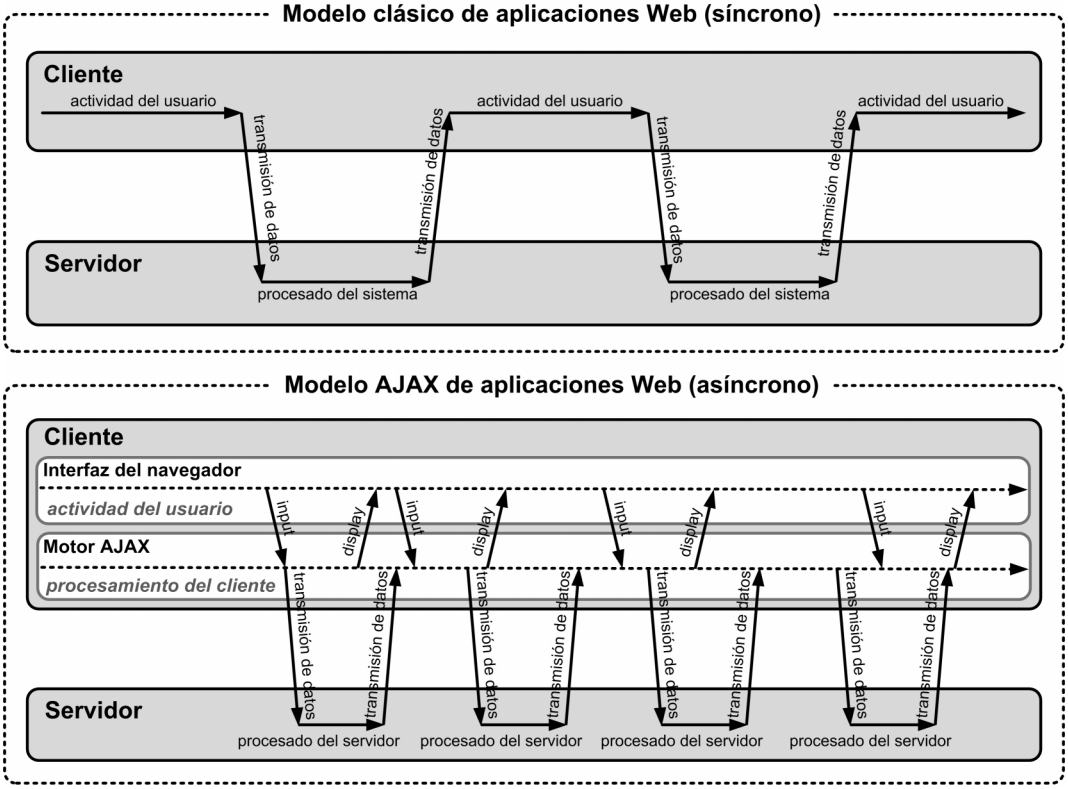


Figura 55.- Arriba.- Cronograma de petición-respuesta en el modelo clásico. Abajo.- Representación de petición-respuesta en el modelo basado en AJAX.

Como ventajas del uso de AJAX destacan una significativa reducción del ancho de banda de la comunicación al haber menos intercambio de información cliente-servidor y la posibilidad de interacción asíncrona entre el cliente y servidor, frente al modelo clásico. Por el contrario, inicialmente es necesario descargarse la página completa, incluyendo el código JavaScript, por lo que es un proceso más lento y el cliente debe soportar parte del procesado de información al ejecutar JavaScript [147].

Gracias al empleo de la tecnología AJAX las peticiones HTTP que el cliente realiza y requieran la intervención del servidor se transforman en peticiones JavaScript que se hacen al elemento encargado de AJAX, de forma asíncrona, evitando que deje de interactuar con la interfaz debido a la recarga de la página por parte del servidor.

Para conseguir tal propósito, AJAX se basa en la combinación de 4 tecnologías: DOM[23] (Document Object Model), un modelo computacional a través del cual las aplicaciones y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de documentos HTML[24] y XML, JavaScript, HTML (HyperText Markup Language) y XML (eXtensible Markup Language), lenguaje de marcas sencillo y flexible que permite representar datos e incluir una descripción de éstos.

[23] EL responsable del modelo DOM es el consorcio W3C.
[24] En conjunción con CCS u hojas de estilo.

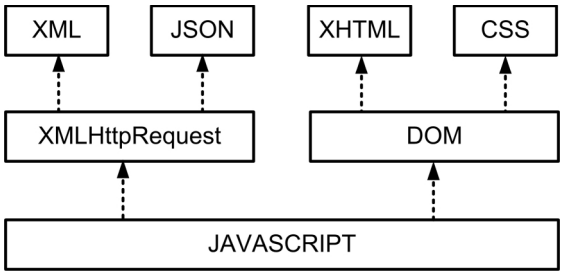


Figura 56.- Representación de tecnologías asociadas a AJAX

5.5.3. Principio de funcionamiento de la comunicación con la interfaz Web

Cuando un usuario desea hacer uso del sistema de instrumentación, previa autentificación en el sistema, se llamaría a la URL de la página Web del laboratorio de instrumentación eDSPlab+, que se encuentra instalada en el servidor[25]. Cuando el cliente remoto interactúe con la página, los eventos producidos sobre la interfaz, que es implementada como un formulario, estarán asociados a funciones JavaScript que se comunicarán con los distintos Servlets del servidor. El servidor ejecutará el *servlet* asociado a la petición del extremo cliente y retornará los datos que se precisen en un formato XML.

En el caso de que el evento que llega al servidor corresponda con un comando a ejecutar sobre un instrumento, el Servlet se encargará de escribir en el búfer de entrada que emplea la aplicación Java del controlador de instrumentación el comando en cuestión y, si fuese preciso recibir una respuesta, leería el búfer de salida para hacer llegar los datos a la aplicación cliente nuevamente.

El servidor, implementado utilizando Tomcat, tendrá una doble misión, por una parte será el responsable de exportar la página Web para que ésta sea accesible por el cliente y, por otra, deberá monitorizar las posibles acciones del cliente remoto sobre la interfaz, llamando y ejecutando los *servlets* asociados a estas acciones que se comunicarán con el controlador remoto del sistema de instrumentación a través de los búferes de entrada/salida.

[25] El servidor Web instalado es Tomcat, el cual implementa las especificaciones de los *servlets* y de JSP de *Sun Microsystems*.

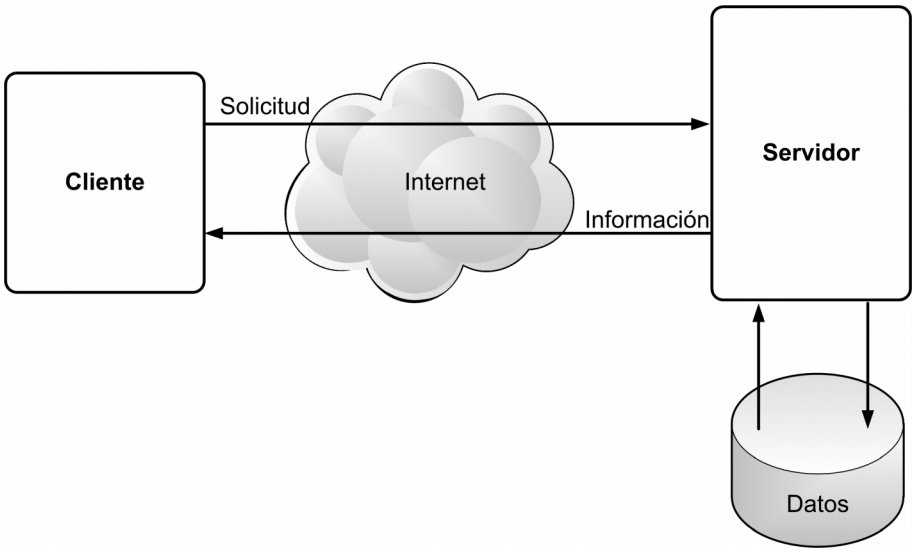


Figura 57.- Estructura del principio de funcionamiento con la interfaz Web

Para el diseño de las páginas Web de los clientes se ha decidido utilizar la tecnología AJAX debido a que esta tecnología nos permite realizar peticiones asíncronas que implican la recarga de zonas concretas de la página Web, no su totalidad, reduciendo el tráfico de red asociado al mismo tiempo que mejora el tiempo de respuesta del servidor.

La interfaz HTML del cliente utilizará hojas de estilo CSS para separar el formato y estilo de las páginas del código HTML propiamente. Cada botón de la interfaz Web asociado a una acción tendrá una correspondencia con el comando contemplado en el archivo "*properties*" asociado al instrumento real sobre el que se quiera actuar. Finalmente, el empleo del lenguaje JavaScript implementará las diversas funcionalidades que dotarán de dinamismo a la interfaz, permitiendo la interacción entre el cliente, la página Web y el controlador de instrumentación remoto.

Para aumentar la versatilidad de cada uno de los bloques que componen la interfaz Web, cada instrumento tiene asociado un conjunto de páginas Web, archivos JavaScript y hojas de estilo CCS independientes unos de otros.

El cliente remoto, a través de la interfaz Web, realizará peticiones HTTP al servidor a través de funciones JavaScript residentes en la página, no directamente. Las funciones JavaScript enviarán la petición y los datos al servidor y éste devolverá los mismos en un formato del tipo XML. Los datos recibidos serán formateados por las funciones JavaScript de la página Web del navegador del cliente y serán mostrados en la pantalla actualizando el árbol de componentes DOM. Es decir, el servidor no genera una nueva página Web, únicamente los datos que se precisan de respuesta[26].

5.5.4. Arquitectura de la interfaz Web del osciloscopio

El diseño de la interfaz del osciloscopio se basa en la división de la consola del mismo en *frames*, cada uno asociado a una funcionalidad distinta.

[26] Este enfoque es lo que denominamos AJAX.

La página HTML principal, contendrá tres *frames*, asociados a la pantalla, la botonera principal y los botones del menú. A su vez, el formato y diseño del mismo se basa en la utilización de una hoja de estilo CSS. Para proveer de mecanismos de comunicación entre el servidor y dicha interfaz se emplean dos funciones JavaScript.

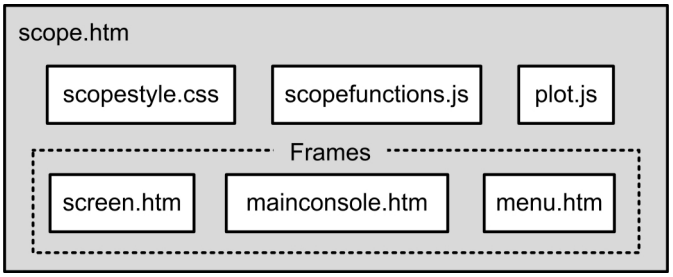


Figura 58.- Estructura de la interfaz Web del osciloscopio

La presentación de la interfaz Web del osciloscopio al usuario se muestra embebida en una página principal, scope.htm, que albergará toda la funcionalidad del osciloscopio. Dicha interfaz se encuentra, a su vez, dividida en tres *frames*, screen.html, menu.htm y mainconsole.htm, asociados a la pantalla de representación, el menú principal del osciloscopio y la consola principal.

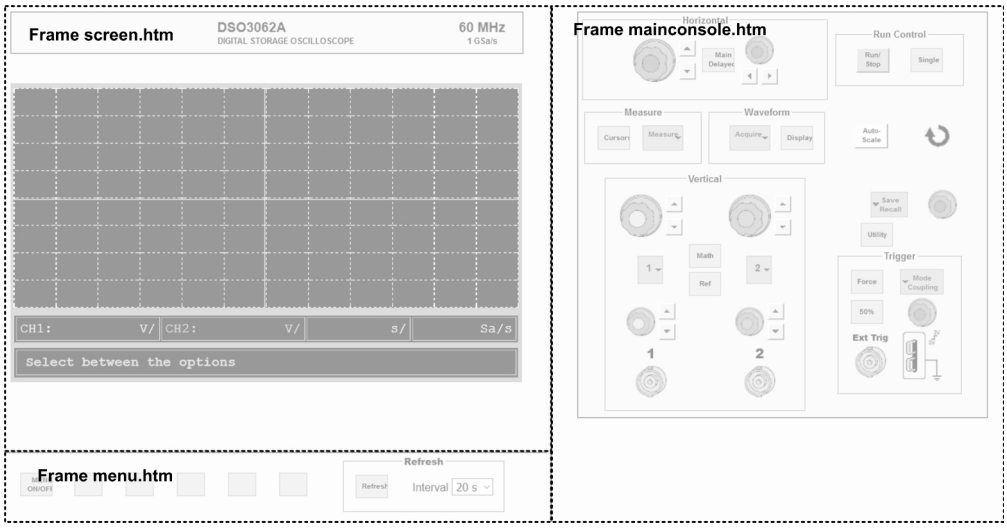


Figura 59.- División en *frames* de la interfaz principal del osciloscopio

Cada *frame* incorporará la etiqueta “name” necesaria para poder acceder a ellos en el lenguaje JavaScript haciendo uso de DOM.

```
<frameset cols="55%,*">
  <frameset rows="85%,*">
     <frame src="screen.htm" scrolling="no" name="screen">
     <frame src="menu.htm" scrolling="auto" name="menu">
  </frameset>
     <frame src="mainconsole.htm" scrolling="auto" name="mainconsole">
  <noframes>
  <p> The Web browser doesn't support frames </p>
  </noframes>
</frameset>
```

Figura 60.- Fragmento del código asociado a los *frames* de scope.htm

Frame screen

El diseño de la pantalla del osciloscopio debe permitir representar las formas de onda solicitadas bajo petición del cliente, bien de forma manual o a través de un refresco periódico. La pantalla debe poder representar ningún canal, uno de los dos canales de forma independiente o ambos a la vez, además de otra información como es la cuadrícula de división temporal y de tensión e información textual sobre medidas, entre otros.

Para hacer esto posible se ha hecho uso de la tecnología de diseño Web basada en capas[27]. Las capas, identificadas mediante etiquetas HTML , no son más que recuadros que pueden ser situados en cualquier parte de la página y en los cuales podemos insertar código HTML; pudiendo ser mostradas y ocultadas según se precise, lo cual es especialmente útil en nuestro caso para, por ejemplo, la gestión de los canales del osciloscopio.

Para la representación gráfica de la pantalla se ha seguido un modelo de cinco capas superpuestas correspondiente al fondo, rejilla, ejes horizontal y vertical, representación de la onda del canal 1 y representación de la onda del canal 2. Las dos últimas capas, correspondientes a las ondas mostradas por el equipo, son generadas de forma dinámica cada vez que se representan, pues deben ser solicitadas al controlador de instrumentación remoto.

[27] Si bien hace unos años las páginas Web basadan la mayoría de sus diseños en el empleo exclusivamente de tablas para organizar la información, actualmente la tendencia es hacer uso de las capas, asociadas a las etiquetas , junto con el uso de tablas y hojas de estilo.

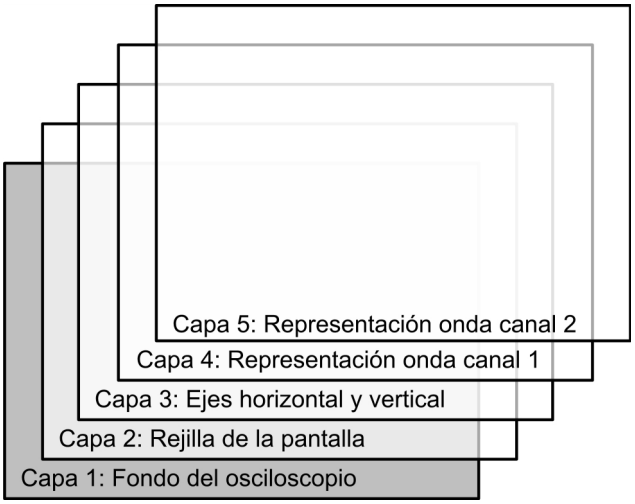


Figura 61.- Estructura de las capas asociadas a la pantalla del osciloscopio

Además, en la parte inferior de la pantalla, se han dispuesto otras dos capas asociadas a unos indicadores de medida textuales y una zona que muestra información sobre el estado del sistema. De tal forma que la consola gráfica del osciloscopio quedaría organizada como se muestra en la siguiente figura.

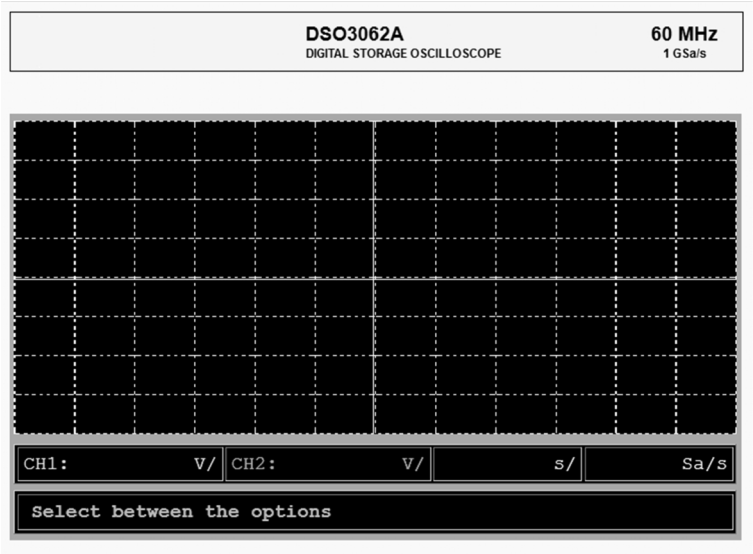


Figura 62.- Estructura de la consola gráfica

Frame mainconsole

El *frame* mainconsole tiene la función de presentar al cliente una interfaz que le permita configurar el osciloscopio y seleccionar la forma en la que desea que se represente la información. Para ello se han dispuesto diferentes pulsadores, botones y selectores asociados cada uno de ellos a una función o comando que será enviado al controlador de instrumentación en caso de interactuar con ellos.

Los distintos controles de la interfaz han sido agrupados por funcionalidad empleando los “fieldsets” (grupos de campos) que el lenguaje

HTML pone a nuestra disposición, permitiendo mostrar una leyenda asociada a los mismos.

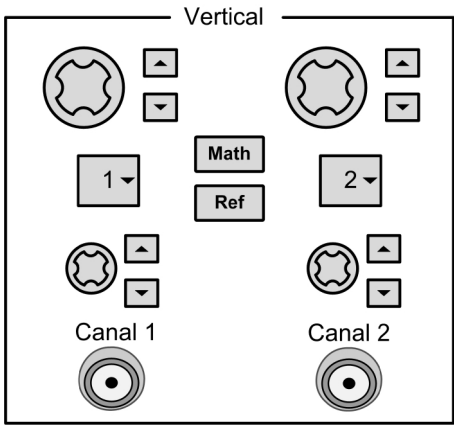


Figura 63.- Ejemplo de leyenda asociada a un conjunto de controles agrupados con la etiqueta "fieldset"

Cada control o botón de la consola se implementa con la etiqueta "input" empleada en los formularios HTML y dispone, a su vez, de una serie de campos o atributos que son necesarios para interactuar con éstos y que precisaremos para la comunicación con el controlador de instrumentación remota.

Atributo	Descripción
type	Especifica el tipo de control que se desea crear. En el osciloscopio serán de tipo "button"
class	El atributo "class" determina la apariencia, previamente definida en la hoja de estilo referenciada.
id	Permite asignar un nombre al elemento. Se emplea, entre otras posibilidades, como medio de hacer referencia a un elemento en concreto desde un lenguaje Script, como JavaScript.
name	Especifica un nombre para el elemento input. Definirá la función del botón, al estar asociada directamente con el comando especificado en el archivo "properties" del controlador de instrumentación.
value	Es el texto que se verá visualmente sobre el botón en la interfaz cliente

Tabla 4.- Atributos de la etiqueta "input" de HTML

Dos atributos especialmente importantes serán el atributo "id", un identificador único que nos permite acceder al botón desde JavaScript, y el campo "name", que sirve para indicar la funcionalidad del botón, existiendo una correspodencia entre el nombre de este campo y el comando real que se ejecutará en el instrumento[28].

[28] El botón "RUN", por ejemplo, cuyo name es "run", estará asociado al comando ":RUN" del archivo "properties" correspondiente del servidor, en otras palabras, existirá la línea "run = :RUN" en dicho documento.

Para reducir el número de botones que aparecen visualmente en la consola, se han implementado algunas funcionalidades mediante menús desplegables.

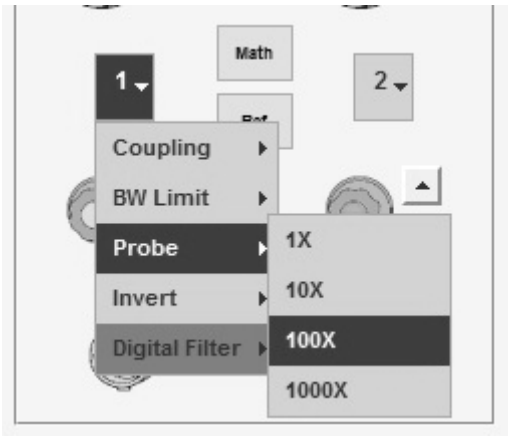


Figura 64.- Detalle de menú desplegable para configuración de la sonda

Frame menu

Este *frame* implementa las funcionalidades del menú principal del osciloscopio, de forma similar a como se implementan en el *frame* mainconsole.html e incorpora además las funcionalidades de refresco de la pantalla del osciloscopio, los indicadores de la pantalla y la consola de información. A través de las opciones de refresco de este menú se puede solicitar el refresco manual de la pantalla del osciloscopio o bien seleccionar un tiempo de refresco periódico que actualice la pantalla de forma temporizada.

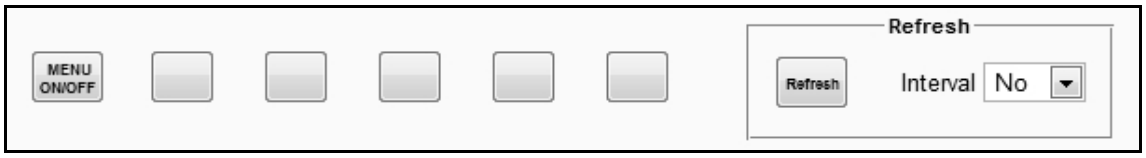


Figura 65.- Detalle del menú del *frame* menu.htm

Proceso de comunicación con el servidor

El proceso de comunicación con el servidor se basa en la tecnología AJAX, como ya se ha comentado con anterioridad, cuyo objetivo es convertir la pulsación de los botones en peticiones de actuación sobre el sistema de instrumentación remota del servidor.

Para que lo anterior sea posible, el documento HTML principal, mediante la etiqueta <Script>, enlazará los archivos que contienen las funciones JavaScript necesarias para invocar los Servlets del extremo del servidor.

```
<script src="./js/scopefunctions.js"
 language="JavaScript"></script>
```

Figura 66.- Enlazado de las funciones JavaScript

En primer lugar, cuando se produce la carga de la página principal HTML se invoca a la función JavaScript “addEvent”, a la que se le pasará como parámetro otra función, “buttonEvents”, responsable de registrar todos los eventos de pulsación de botones de la consola.

Dicha función recorre todos los botones y menús desplegables, capturando su identificador y llamando a la función “addEvent” por cada elemento, a la que le pasa como parámetros la referencia del botón, el tipo de evento al que debe reaccionar (click) y la función que ejecutará, “pushButton”.

La función “pushButton” será la encargada de implementar la funcionalidad propiamente del botón al ser presionado y construir la URL que será enviada al Servlet del servidor remoto. Dicha función comprobará si el botón pulsado se corresponde con un comando a enviar, en caso negativo se producirá un mensaje por pantalla. En el caso que la pulsación corresponda con un comando, se recupera el atributo “name” del botón e invoca a varias funciones que, partiendo de la URL del servidor remoto, le añade una cabecera con un número aleatorio, para evitar problemas de caché y otra cabecera con el identificador del instrumento y el atributo “name” del botón pulsado.

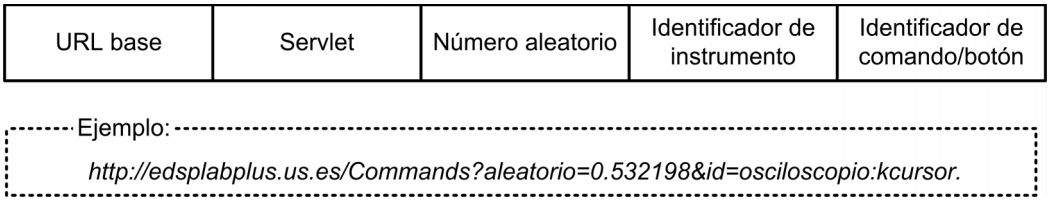


Figura 67.- Estructura de cabeceras de la URL enviada al servidor

A continuación se llamaría a una función “loadURL”, dicha función captura como parámetro la nueva URL generada; que será la que se emplee para hacer la petición al servidor. A continuación crea un objeto XMLHttpRequest, que será el responsable de la comunicación asíncrona con el servidor y estará asociado a la conexión que deseamos establecer. Se prepara la posible respuesta del servidor (caso de recibir datos del mismo), empleando el método open(), el cual prepara una conexión HTTP a través del objeto XMLHttpRequest, con el método[29] y URL indicados como parámetros (la URL será la definida anteriormente) y de forma síncrona o asíncrona, según le indiquemos (asíncrona en nuestro caso). Finalmente se invoca el método send(), que envía la petición al servidor propiamente.

[29] Existen dos posibles métodos, GET y POST.

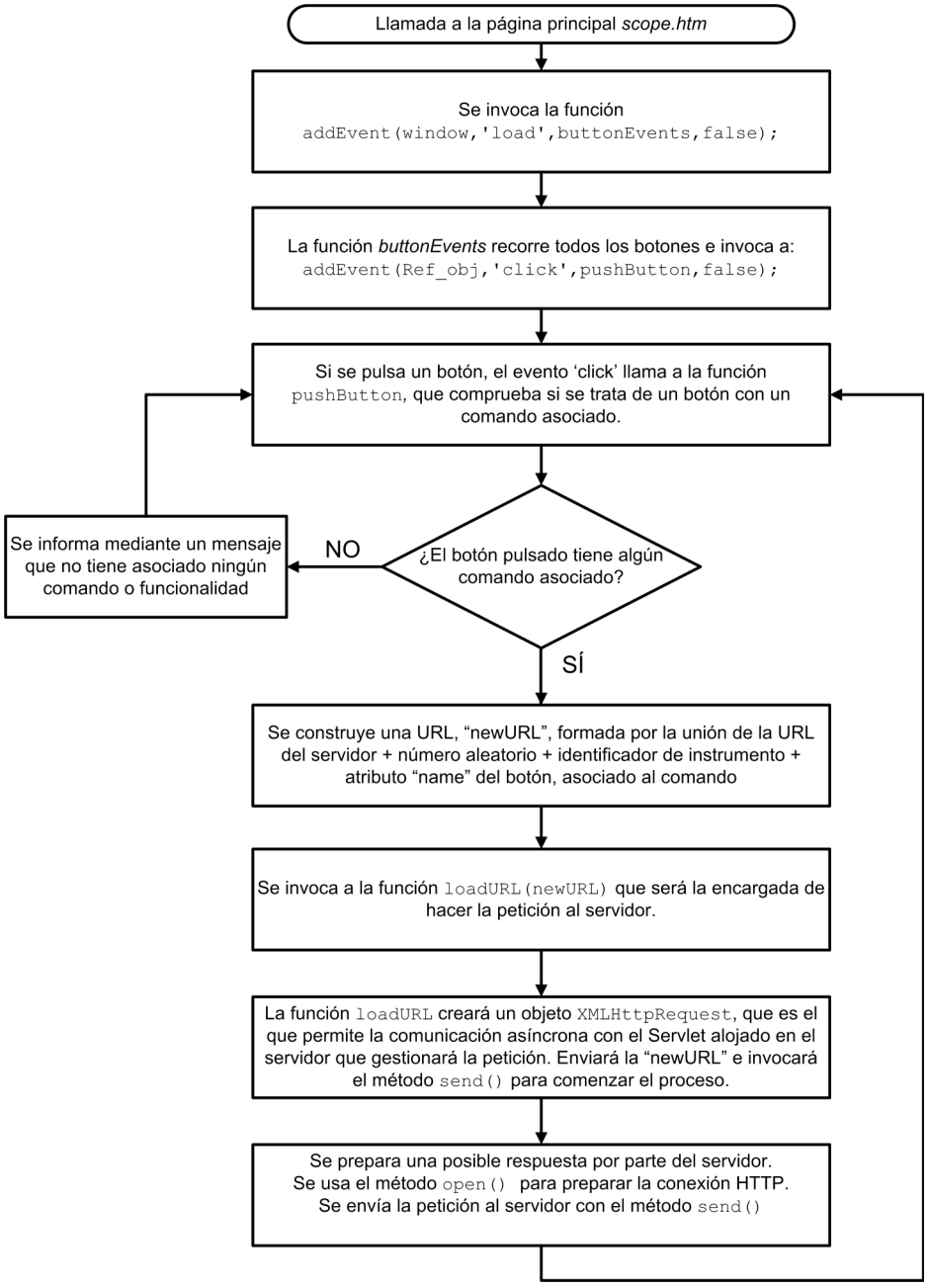


Figura 68.- Diagrama de flujo simplificado del proceso de petición de acción mediante XMLHttpRequest

El objeto de la clase XMLHttpRequest asociado a la conexión establecida nos puede proporcionar información sobre el estado de la misma accediendo a la propiedad ReddyState del mismo. Los posibles estados en los que el objeto se puede encontrar son los que se muestran en la siguiente tabla.

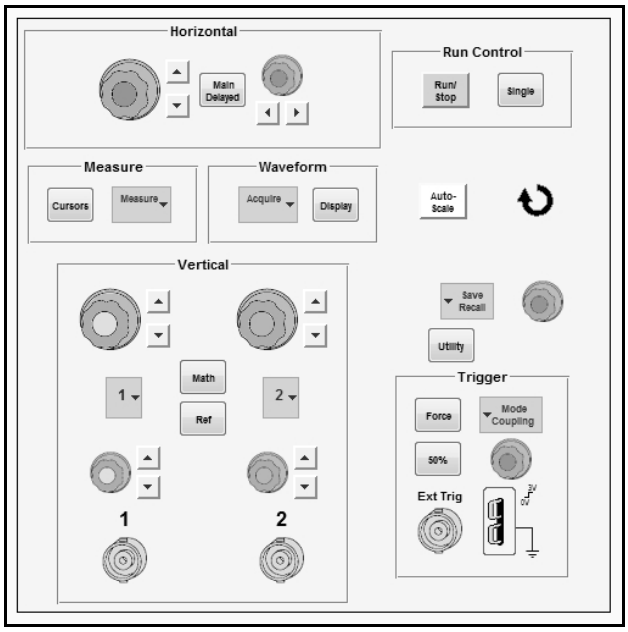
readyState	Estado
0	No inicializado
1	Cargando
2	Cargado
3	Interactivo
4	Completado

Tabla 5.- Posibles estados del objeto XMLHttpRequest

Cada vez que el objeto asociado a la conexión cambia de estado se ejecuta una nueva función. En el caso de que la propiedad readyState adquiera el valor “4” esto indicará que todos los datos han llegado al servidor y mediante el empleo del método “responseText” será posible recuperar la información enviada por el servidor.

Cabe destacar que la mayoría de los comandos asociados a los botones no generan respuesta alguna por parte del instrumento remoto; si bien es posible que se precise actualizar, únicamente, la pantalla del osciloscopio. En los casos en los que no se precise respuesta por parte del servidor, el Servlet asociado enviará una cadena informando que no existe respuesta.

En función del comando se actualizarán aquellas secciones que sean requeridas, como las ondas de los canales de entrada en la pantalla del osciloscopio, por ejemplo, seleccionando en cada momento la información precisa que sea necesario actualizar, reduciendo así el ancho de banda ocupado y la carga de CPU y memoria del dispositivo cliente.



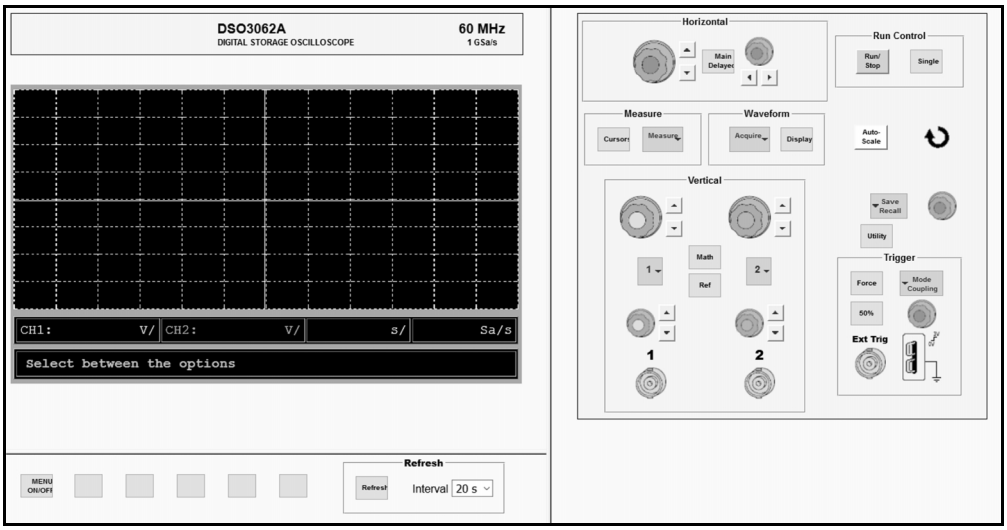


Figura 69.- Consola principal del osciloscopio

5.5.5. Arquitectura de las interfaces Web de la fuente de tensión y generador de funciones

El diseño de la interfaz de la fuente de alimentación sigue la misma estructura que la explicada en el osciloscopio, estando basada en la división de la consola del mismo en *frames*, cada uno asociado a una funcionalidad distinta.

En este caso se ha optado por una página HTML principal que alberga cuatro *frames*, asociados a la pantalla, la botonera principal, los botones para introducir la tensión e intensidad y los conectores. Se emplean hojas de estilo CSS para preservar un formato y diseño homogéneos.

Los mecanismos de comunicación entre el servidor y la interfaz siguen el mismo proceso descrito en la interfaz del osciloscopio, basándonos en AJAX; reutilizando gran parte del código ya implementado.

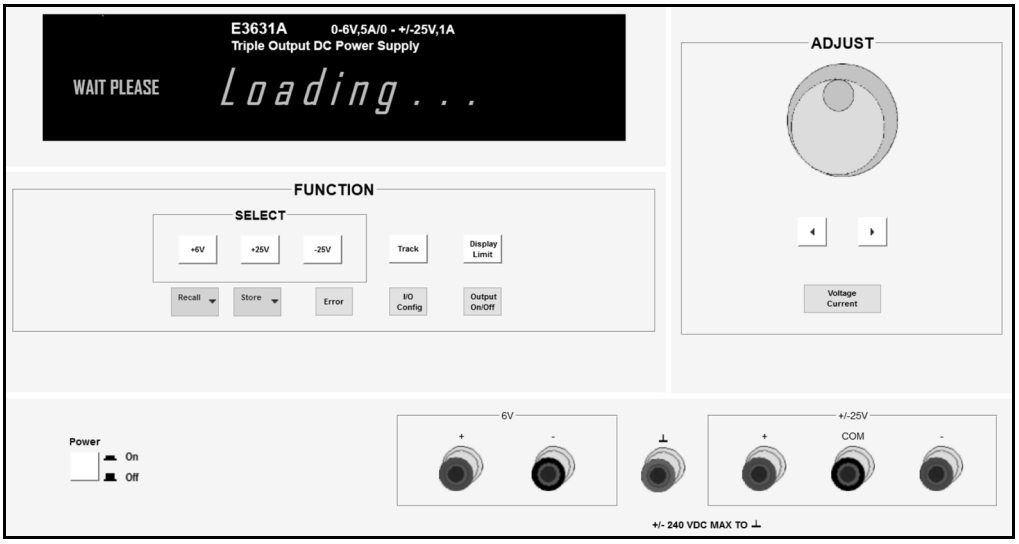


Figura 70.- Detalle de la interfaz Web de la fuente de tensión

Por otra parte, todo lo comentado anteriormente es extrapolable a la interfaz gráfica del generador de funciones, dividido en tres *frames*, uno asociado a la pantalla, otro al equipad y finalmente el panel de funciones.

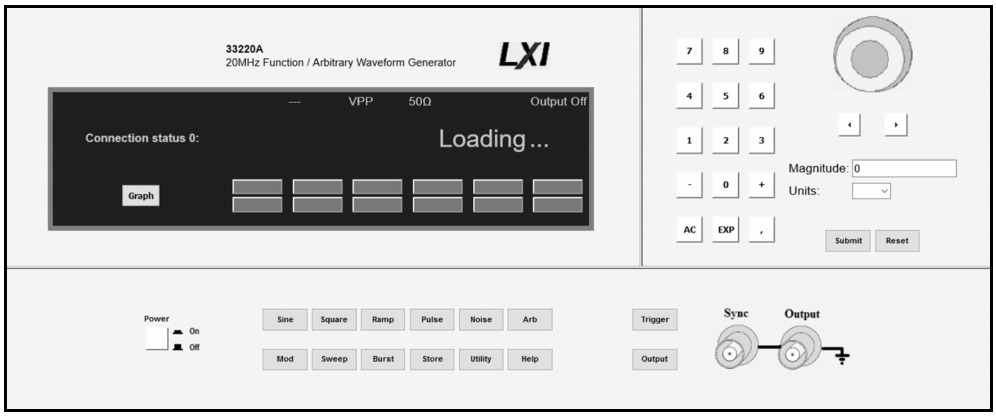


Figura 71.- Interfaz gráfica del generador.

5.5.6. Arquitectura de la interfaz Web del DSK

El diseño de la interfaz Web del DSK implementa algunas de las funciones más importantes que nos ofrece el entorno de programación Code Composer Studio.

Función	Descripción
Load	Función empleada para cargar en memoria del DSP un proyecto o fichero *.out
Open	Se emplea para abrir un proyecto creado
Build	Opción responsable de “construir” el proyecto a lenguaje máquina
Rebuild	Recompilado de la opción Build
Halt	Opción para parar la ejecución del proyecto cargado en memoria
Run	Se emplea para ejecutar el proyecto que se encuentra en la memoria tras haber sido cargado con “Load”.

Tabla 6.- Relación de funciones implementadas en la interfaz Web del DSK

La estructura seguida para el desarrollo de la interfaz Web que simula el entorno Code Composer se compone de un único documento HTML donde se incrustan las hojas de estilo necesarias para gestionar la apariencia y los menús desplegables de cada una de las opciones implementadas y un fichero con las diferentes funciones JavaScript que serán empleadas.

Dentro del entorno se ha incrustado la gestión de subida de los ficheros asociados al proyecto que se desea volcar y ejecutar en la memoria del DSP. Para ello se ha empleado un formulario donde el campo “action” se ha asociado al Servlet del servidor remoto encargado de dicha tarea. Mediante un campo “input file”; el formulario permite la subida de ficheros al servidor, habiéndose habilitado la posibilidad de subir un total de seis archivos como máximo. Al pulsar sobre la opción de envío, el formulario enviará el fichero o ficheros para ser alojados en el servidor.

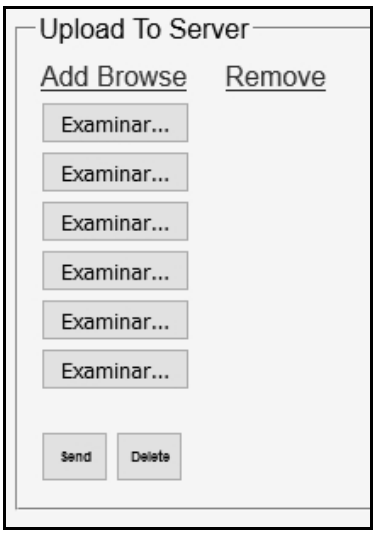


Figura 72.- Detalle de la interfaz de subida de archivos

Otra de las funcionalidades que presenta la interfaz es la presentación de una sección “Log” que nos informa del estado y procesos de la aplicación en el servidor, Para ello es necesario acceder al contenido de dicha información el el servidor local, dicha información se encuentra en los archivos cc_build_DEbug.log y cc_build_Release.log de la herramienta Code Composer, disponibles en el servidor; de tal manera que la información de ejecución sea accesible desde la interfaz cliente.

Para poder recibir respuestas por parte del servidor frente a acciones del cliente final sobre la interfaz Web; como la carga en el DSK de un proyecto previamente subido en el servidor por el usuario, se ha habilitado a la interfaz de un mecanismo de comunicación con el servidor basado en el uso de un “iframe”, que no es más que un elemento HTML que nos permite incrustar dentro de la página principal HTML otro documento HTML que será construido dinámicamente de forma remota por el servidor. Empleando el atributo *target* de un enlace HTML o el lenguaje JavaScript, es posible sustituir el documento incrustado por otro distinto sin necesidad de volver a cargar la página principal, motivo por el cual es también empleado en las aplicaciones AJAX como alternativa a XMLHttpRequest. Así, por ejemplo, frente a la acción de pulsación “Load” en la interfaz Web del cliente, que cargará el proyecto seleccionado sobre la tarjeta DSK, el servidor responderá con el mensaje “Transferencia Correcta”, si el proceso ha tenido lugar exitosamente.

El proceso de comunicación con el servidor es similar al descrito en el resto de interfaces. La pulsación de un botón del interfaz desencadenará la ejecución de un conjunto de funciones JavaScript que generarán una URL válida que invocará al Servlet responsable en el extremo del servidor.

Otro aspecto importante es el relativo a la gestión de los archivos subidos de los proyectos por parte del usuario, lo cual ha sido integrado dentro de la interfaz, como ya se ha comentado, para lo cual el usuario se habrá autentificado previamente y se le habrá asignado una carpeta o directorio en exclusiva dentro del servidor.

Por otra parte, la sección inferior de la interfaz se ha dotado de una sección “Log”, correspondiente a un campo de texto HTML (tipo “textarea”) que será empleada tanto para volcar la información recibida por parte del servidor tras el envío de un comando como las operaciones que el propio usuario de la interfaz realice sobre la misma.

5.5.7. Seguridad en el acceso de los usuarios

La gestión de un acceso restringido por parte de los usuarios remotos en el modelo propuesto en eDSPlab+ resulta relativamente sencilla al utilizar tecnologías Web; las cuales ofrecen una amplia versatilidad en cuanto a la forma y complejidad de la solución a adoptar, así como en la compartición de los recursos.

Se ha implementado una sencilla solución basada en la autentificación de usuarios previamente registrados en la aplicación por medio de una petición por correo electrónico, a los cuales se les proporciona un usuario y clave de acceso al laboratorio de instrumentación remota. Cuando el usuario accede al laboratorio se le solicitará los datos de autentificación y la información recogida se hará llegar a un *servlet* en el servidor que accederá a una base de datos[30] y determinará si el usuario está registrado y los permisos que éste dispone para acceder a las funcionalidades de la aplicación Web.

Figura 73.- Captura de pantalla del acceso al sistema de instrumentación

5.6. Servidor Web

La interfaz Web asociada a los clientes se basa en el diseño de páginas HTML que emplean la tecnología AJAX para implementar un canal de comunicación bidireccional asíncrono entre el cliente y el servidor. Dicho canal permite que, cuando el usuario actúa sobre un control se informe al servidor Web del comando o acción realizada y se comunique éste con los búferes de entrada-salida del sistema de instrumentación o del sistema de desarrollo, según sea el caso, pudiendo recibir una respuesta por parte del servidor y actualizar en la interfaz Web únicamente aquella parte de la misma que haya

[30] Se ha utilizado PostgreSQL como sistema de gestión de base de datos

sufrido alguna modificación; caso de la actualización de la pantalla del osciloscopio, por ejemplo.

Para la implementación de las aplicaciones responsables en el extremo del servidor de recibir las peticiones de los clientes, comunicarse con el controlador del sistema de instrumentación vía búferes, recibir la respuesta del mismo y remitirla a la interfaz, entre otras funciones, se ha decidido emplear la tecnología JAVA basada en los Servlets [144], [145].

La funcionalidad total del servidor se ha implementado dividiéndola en tres bloques claramente diferenciados, cada uno correspondiente a un Servlet que se ejecuta en el servidor. Un primer bloque, será el encargado atender las peticiones que el cliente Web realiza sobre el osciloscopio, el generador de funciones y la fuente de alimentación, es decir, sobre la instrumentación propiamente. En segundo lugar tenemos un segundo bloque responsable de gestionar la interfaz con el DSK en exclusiva, siendo responsable tanto de la subida de ficheros asociados a los proyectos en una carpeta asociada al usuario, previamente autentificado, como de hacer llegar la petición de comandos por parte del cliente Web al software Code Composer Studio, a través del búfer de entrada asociado, que será leído por el extremo del controlador. Por último, disponemos de un tercer bloque que se encarga de gestionar la autentificación de los usuarios que deseen hacer uso de eDSPlab+, comunicándose con la base de datos existente en el servidor. Este bloque también permitirá crear nuevos usuarios a través de la autentificación de un usuario administrador.

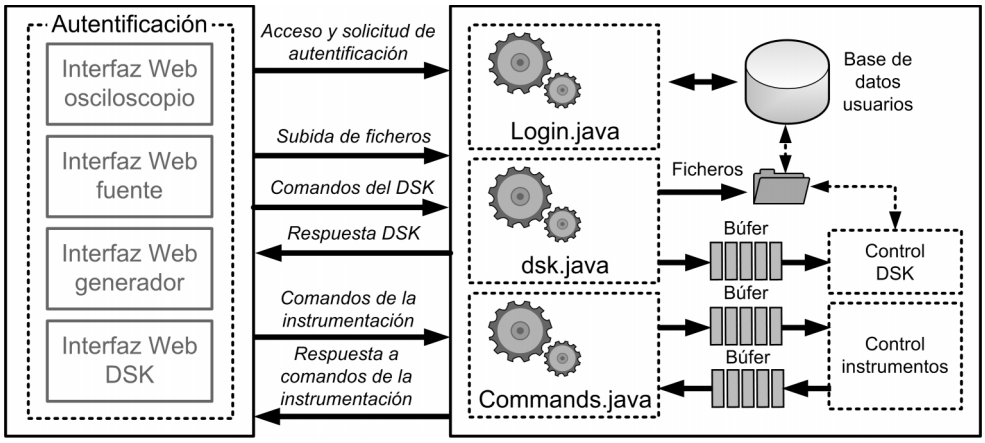


Figura 74.- Esquema de funcionamiento del servidor Web del laboratorio de instrumentación

5.6.1. Servidor de peticiones al sistema de instrumentación

Para atender las peticiones asociadas a los comandos enviados a la instrumentación remota se ha creado un Servlet, denominado “Commands.java”, responsable de recoger los eventos generados por el cliente sobre la interfaz Web de los distintos instrumentos y hacérsela llegar al controlador de instrumentación a través de los búferes de entrada. A su vez, dicho Servlet gestionará la recogida de una posible información de respuesta y la remitirá al cliente Web que realizó la petición.

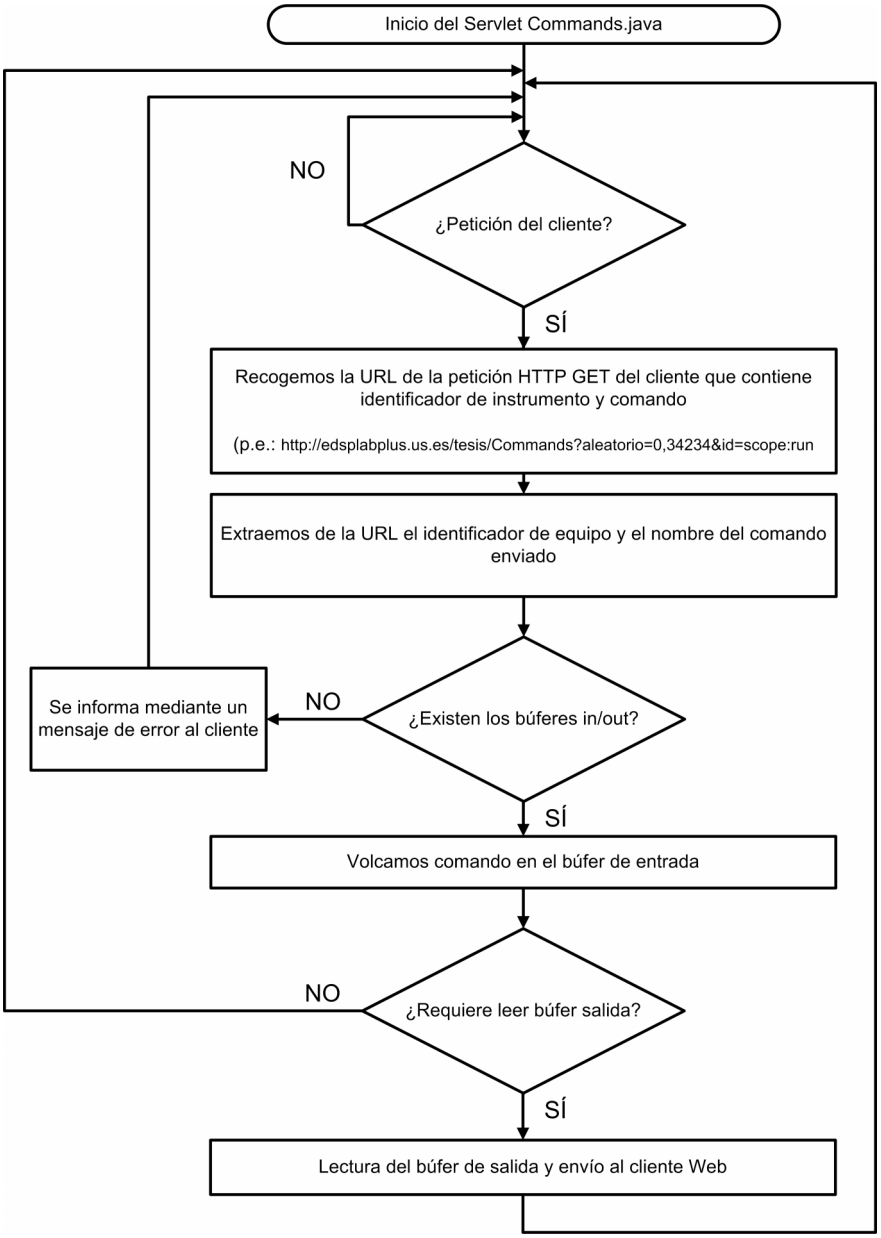


Figura 75.- Diagrama de flujo del servidor de peticiones al sistema de instrumentación

5.6.2. Servidor de peticiones al sistema DSK

La gestión del DSK se ha separado de la gestión del sistema de instrumentación para favorecer la modularidad de la solución adoptada.

En este caso tenemos dos posibles tipos de petición que podemos hacer al Servlet “dsk.java”, encargado de la gestión del DSK. Las peticiones asociadas a la petición de ejecución de comandos en el Code Composer Studio tienen la misma estructura que en el caso del Servlet “Commands.java”, estando implementadas utilizando peticiones HTTP GET, mediante un método “doGET()” [144] y accediendo a un búfer de entrada exclusivo para la comunicación con el DSK. Por otra parte, la gestión de la subida de los ficheros del proyecto, implementada mediante un formulario en la interfaz cliente, comprueba la autentificación del usuario, asociándole una carpeta en el servidor para la subida de ficheros, sube los ficheros a dicha carpeta y

devuelve al cliente la cadena “Transferencia correcta”. En este caso se emplea el método “doPOST()”.

5.6.3. Servidor de autentificación

Por último, contamos con un Servlet, login.java, asociado a la autentificación del usuario en el sistema, el cual deberá comunicarse con la base de datos de usuarios. La dirección Web base del laboratorio de instrumentación da acceso a una pantalla inicial que invita al usuario a autentificarse, lo que posibilitará el control del sistema de instrumentación y habilitará la subida de archivos al sistema y control del DSK.

En este bloque también se permitirá la creación de nuevos usuarios de la plataforma, mediante la autentificación como administrador del sitio.

5.7. Discusión y conclusiones

Las herramientas que se han empleado en el modelo propuesto, así como los lenguajes de programación, son de software libre[31], quedando de manifiesto que es posible diseñar sistemas de instrumentación remota sin la necesidad de emplear licencias o software especializado de pago.

La solución adoptada, al emplear lenguaje, es portable, y puede ser empleada en otras plataformas y sistemas operativos. Esta portabilidad únicamente se ve comprometida en el acceso a los drivers necesarios para controlar la comunicación con el bus de instrumentación, que depende del SO operativo y arquitectura procesadora del servidor. Sin embargo, dado el diseño modular y gracias al empleo del Java Native Interface, bastaría con diseñar una nueva capa de acceso al bus para la plataforma en cuestión, siendo el resto de elementos funcionalmente idénticos.

Gracias a la utilización de búferes de entrada-salida en la comunicación entre cliente y servidor, se resuelve la independencia entre la tecnología empleada en el diseño de la interfaz gráfica necesaria en el cliente y la aplicación que se ejecuta en el servidor, lo que contribuye a una mejora en la modularidad y escalabilidad del sistema.

El modelo no implica la instalación de ningún plug-in especial en el cliente, lo que implica la posibilidad de trabajar con clientes delgados con muy reducida necesidad de procesamiento y dispositivos dotados de navegadores Web compatibles con JavaScript y HTML, incluido smartphones y tablets.

[31] A excepción de la plataforma Code Composer Studio.

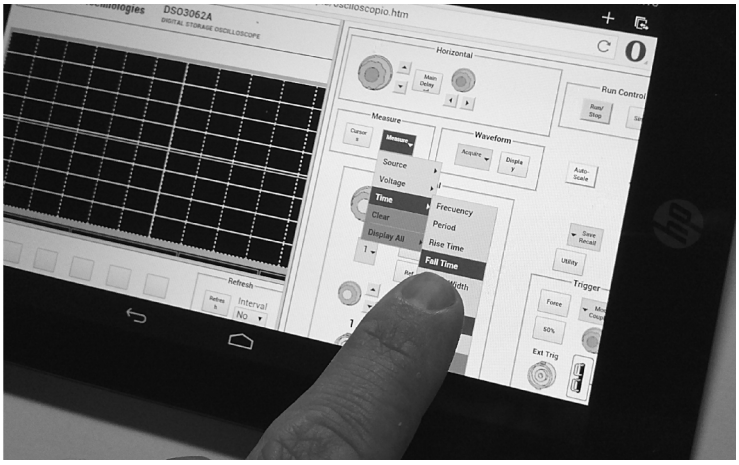


Figura 76.- Vista de la consola del osciloscopio sobre tablet HP Slate7 Plus

El uso de Java nos permite incorporar utilidades como Javadoc, una utilidad de Oracle que se emplea para la generación de documentación de APIs en formato HTML a partir de código fuente Java, además de la creación de consolas locales de actuación y monitorización del sistema servidor con una versatilidad elevada.

La utilización de AJAX en el diseño de las páginas Web del cliente supone la posibilidad de realizar peticiones asíncronas que permiten reducir el ancho de banda del tráfico de información, al recargarse zonas concretas de la Web, mejorando el tiempo de respuesta. Además, el conjunto de tecnologías que conforman AJAX permiten aumentar la interactividad con la página Web, desconectando las acciones de los clientes del envío de datos al servidor. AJAX evita que se tenga que actualizar la totalidad de la página Web al convertir las peticiones HTTP en datos devueltos en formatos tipo XML, que son gestionados por las funciones JavaScript de la página del cliente y que son los encargados de actualizar la interfaz Web que se presenta al usuario final, todo ello en el marco del navegador Web. Además, la utilización de AJAX no precisa la instalación de ningún plug-in en el cliente, únicamente que el navegador permita la ejecución de JavaScript.

La utilización de tecnologías Web nos permite añadir multitud de prestaciones adicionales al laboratorio, como son la inclusión de archivos “log” que informen sobre las acciones de los usuarios, archivos subidos, etc.

También es posible integrar el diseño dentro de un único entorno, gestionando autentificaciones y sesiones de usuarios, de tal forma que sea muy complejo acceder al mismo tecleando, sencillamente, una dirección Web. La integración a la que se hace referencia queda manifiesta, por ejemplo, al observar el proceso de subida del proyecto a volcar sobre el DSP, que se encuentra embebida dentro de la interfaz Web del entorno que simula a Code Composer Studio, no siendo preciso crear un servidor paralelo para dar tal servicio. Sumado a lo anterior, el uso de tecnologías Web permite integrar la gestión del recurso y los usuarios dentro de un LMS o un CMS, compartiendo la base de datos y simplificando el proceso de autentificación, entre otras mejoras.

Sin embargo, el actual modelo presenta algunas limitaciones que pueden ser fruto de futuras líneas de investigación. En primer lugar, nos hemos centrado únicamente en la instrumentación GPIB, por ser el sistema de instrumentación del que se disponía originalmente. Por otra parte, no se ha dotado al sistema de mecanismos de creación y descubrimiento de nuevos instrumentos.

También debemos destacar que si bien el empleo del programa Code Composer Studio y la aplicación Ccs_scripting que ofrece desde la versión 2.1 simplifica el proceso de interacción con el usuario remoto, esto implica una dependencia con la plataforma de programación y el sistema operativo Windows, lo que hace que este módulo del sistema no sea multiplataforma ni totalmente abierto, actualmente.

Por otra parte, también es conveniente considerar que, en la actualidad, sigue existiendo cierta problemática en relación a la compatibilidad de los distintos navegadores a la hora de interpretar el código HTML, debido a que algunas firmas no siguen los estándares sugeridos por el W3C, lo que implica un sobreesfuerzo de programación en muchos casos, lo cual resta eficiencia a la solución.

6. Análisis de prestaciones

Si bien podemos encontrar en la bibliografía científica multitud de propuestas de laboratorios y sistemas de instrumentación electrónica; éstas raramente proponen medidas, parámetros o técnicas que nos permitan validar la "calidad de servicio" que dichos sistemas ofrecen.

Entre los principales métodos de "evaluación de la calidad" que podemos encontrar, tenemos aquellos que se basan en encuestas que son entregadas a los usuarios que han empleado el sistema; evaluando la satisfacción en base a los resultados de dichas encuestas, como encontramos en [153], por ejemplo, o los que basan la virtud de las mismas en base a los resultados de su uso, como es el caso de los laboratorios de instrumentación empleados en el ámbito docente, donde se puede establecer una correlación entre la mejora en las calificaciones obtenidas o el grado de deserción de las mismas y el uso del laboratorio de instrumentación.

En la presente tesis se ha perseguido poder evaluar algunos parámetros que, desde un punto de vista objetivo y subjetivo, nos permitan establecer la bondad del modelo de sistema de instrumentación eDSPlab+ propuesto; comparándolo con el modelo de referencia basado en LabVIEW, eDSPlab.

El término "calidad de servicio" tiene diferentes interpretaciones en función del sector o campo de aplicación donde se defina. Así, en las redes de telecomunicación, como las redes telefónicas y de computación, el término calidad de servicio hace referencia al rendimiento percibido por los usuarios de la red. Dicha percepción puede ser objetivamente medida en base a un conjunto de parámetros clave, tales como el ancho de banda disponible, el rendimiento del sistema, el retraso de la transmisión, la disponibilidad, el Ritter, las tasas de error, etc., que pueden ser cuantizables [154].

Por otra parte, los distintos organismos internaciones también atribuyen diferentes definiciones al término; la ISO define la "calidad" como un "*conjunto de características de un elemento que le confieren la aptitud para satisfacer necesidades explícitas e implícitas*" [155]. Por otra parte, la ITU-T la define como "*el efecto global de las prestaciones de los servicios que determinan el grado de satisfacción de un usuario al utilizar dicho servicio*" [156]. Para las aplicaciones en Internet, la Calidad de Servicio es definida por la IEFT como "*la capacidad de segmentar el tráfico o diferenciar entre los tipos de tráficos con el fin de realizarles un tratamiento diferente a los flujos de señal*" [157].

El concepto de calidad de servicio, por tanto, puede tener diferentes interpretaciones, como las referidas a los mecanismos de reserva de recursos en redes de paquetes, por ejemplo. Sin embargo, el término "Calidad de Servicio" también puede referirse al nivel de calidad del Servicio, es decir, a la calidad garantizada en un sistema. Esta definición de QoS, empleada especialmente en los sistemas de telefonía y servicios de *streaming*, entre otros [158], es una métrica que refleja o predice la calidad experimentada de manera subjetiva por el usuario del servicio, expresada frecuentemente en términos de prestación percibida por el usuario final o grado de satisfacción del usuario, como es el caso del MOS (*Mean Opinion Score*), originalmente

estandarizado por la ITU-T P.800 y que surge como un test empleado durante décadas en los sistemas de telefonía para obtener una medida cuantitativa de la calidad del servicio[32] y que actualmente se sigue utilizando en sistemas de telefonía IP y otros servicios [158].

Otro término relacionado con la calidad del servicio es la QoE (*Quality of Experience*), que puede definirse como la aceptabilidad total de una aplicación o servicio percibida subjetivamente por un usuario final; lo que incluye todos los elementos intervinientes en el sistema extremo a extremo, es decir, el cliente, la interfaz, la red, la provisión de servicios, la infraestructura, etc; pudiendo verse afectada, a su vez, por las expectativas de los usuarios y el contexto.

Por tanto, la principal distinción entre QoE y QoS la encontramos en la perspectiva desde la que se desee analizar cada una; es decir, desde el punto de vista del usuario final en el caso de la QoE y desde la perspectiva de las prestaciones de la red para el caso de la QoS.

La QoE, por tanto, está centrada en el usuario, e indica el grado en que el sistema satisface las necesidades del usuario final, siendo una medida extremo a extremo de las prestaciones del sistema, centrada en la capa de servicio.

Para evaluar la calidad de un sistema en base a la QoE podemos atender a dos posibles aproximaciones, fundamentalmente [159], una basada en evaluaciones subjetivas de los usuarios, los cuales ponderan el sistema acorde a su percepción personal de la calidad del servicio[33], y otra basada en métricas objetivas.

Además, la evaluación subjetiva de la QoE puede ser cualitativa o cuantitativa. Las evaluaciones cualitativas se centran en determinar la calidad atendiendo a factores sociológicos, mientras que la evaluación cuantitativa persigue ponderar la calidad en base a una escala numérica, lo que denominamos MOS (*Mean Opinion Score*).

Si bien existen estándares para la medición cuantitativa de la QoE en sistemas, como las aplicaciones de vídeo, donde disponemos de la recomendación ITU-R BT.500-11, o en la calidad de la señal telefónica, con la recomendación ITU-T P.800, donde se indican parámetros a medir, experiencias a seguir y procedimientos que realizar, en la actualidad los sistemas de instrumentación remota no cuentan con ningún estándar o recomendación en relación a la forma de evaluar la QoE.

[32] En el MOS la calidad de la voz es cuantificada y calificada en una escala del 1 al 5; donde 1 significa muy mala calidad y 5 excelente calidad. Los valores del MOS se obtienen mediante un promedio de las opiniones de un elevado número de usuarios.

[33] Una medición de la QoE puede realizarse, por ejemplo, utilizando el valor medio de opinión MOS (*Mean Opinion Score*) de una población de usuarios.

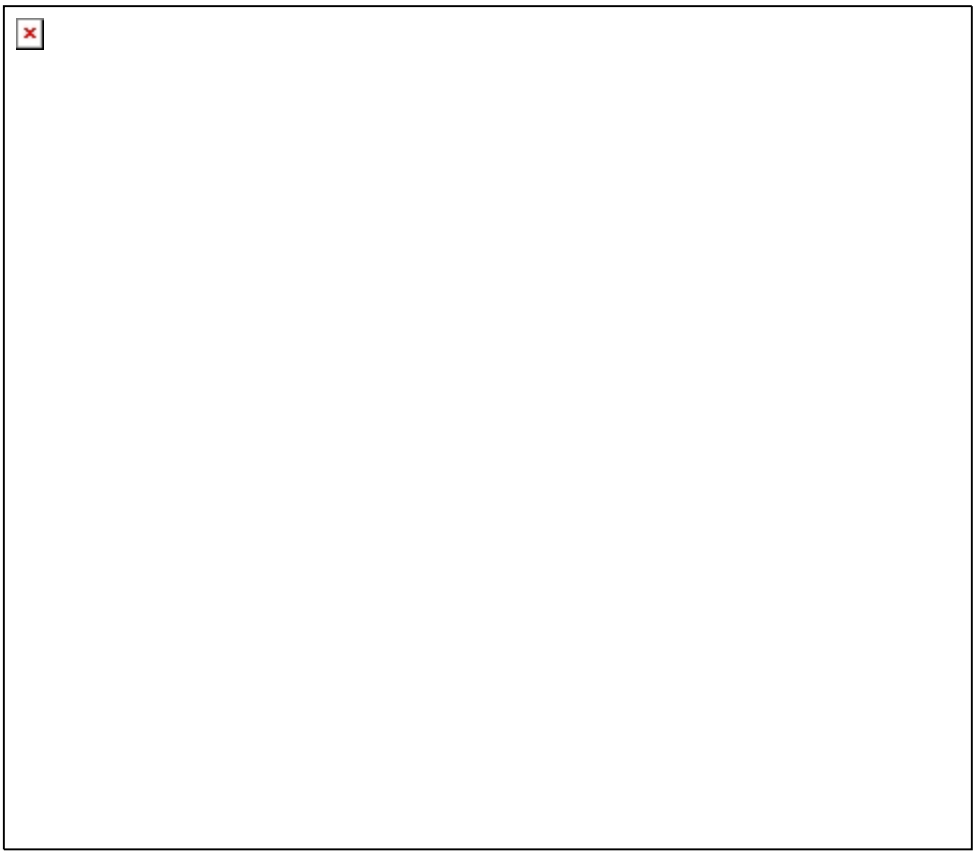


Figura 77.- Ejemplo de instrucciones para participantes de la ITU-T P.800

Con objeto de poder evaluar las prestaciones del modelo de instrumentación propuesto, eDSPlab+, se propone un experimento base a realizar sobre el laboratorio de instrumentación, con un procedimiento a seguir y una serie de parámetros objetivos y subjetivos evaluables desde el punto de vista de la percepción del usuario final; del mismo modo que se procede en otros sistemas como el caso de la telefonía, como hemos visto [160].

6.1. Definición de la experiencia y procedimiento base

Se ha definido un experimento base para comparar las dos implementaciones descritas consistente en la programación y volcado de un filtro FIR paso bajo de cuatro kilohercios sobre el sistema de desarrollo DSK que permitirá monitorizar el efecto del cambio de la frecuencia de una señal de entrada introducida en una entrada analógica del DSP sobre una salida analógica del mismo, estando ambas señales conectadas a los canales 1 y 2 del osciloscopio, respectivamente.

La experiencia partirá de un proyecto que implementa el programa “Filtro FIR” que el usuario remoto deberá subir al servidor, previa autentificación en el mismo, para ello ambos laboratorios se han integrado dentro del contexto de una plataforma LMS tipo Moodle. Una vez cargado el fichero, el usuario accionará la alimentación de la placa DSK a través de la interfaz remota de la fuente de alimentación de la bancada. A continuación seleccionará desde el instrumento remoto el proyecto a cargar en el DSP, comenzando a ejecutarse sobre la placa. El usuario accederá al panel de control remoto del generador de funciones y programará la salida de una onda senoidal que irá variando desde

dos kilohercios hasta seis kilohercios, finalmente ajustará los parámetros necesarios para monitorizar las señales de entrada y salida del DSK en la pantalla de la interfaz del osciloscopio.

6.2. Métricas objetivas de medición de la QoE

Para el análisis y comparación de los sistemas de instrumentación propuestos se han estimado, en primera instancia, un conjunto de métricas objetivas de valoración de la QoE, medidas tanto en el extremo cliente como en el extremo servidor. Dichas métricas son extrapolables a otros sistemas de instrumentación remota, con independencia de la instrumentación que intervenga o el modelo de experimento acometido.

Las métricas objetivas que se han establecido para evaluar y comparar los dos modelos son:

- CPU time.- Parámetro que representa la cantidad de tiempo que los procesos envueltos en el acceso remoto usan la CPU.
- CPU usage.- Porcentaje de tiempo que los procesos utilizan la CPU.
- I/O Writes.- Número de operaciones de escritura de entrada/salida generada por el acceso remoto, incluyendo ficheros, red y dispositivos de entrada/salida.
- I/O Reads.- Número de operaciones de lectura de entrada/salida generada por el acceso remoto, incluyendo ficheros, red y dispositivos de entrada/salida.
- Bandwidth.- Ancho de banda ocupado por la conexión.

Estas métricas afectan directamente a la forma en la que el usuario final percibe la rapidez con la que el sistema de instrumentación responde; pues se basan en estimar la carga de CPU y ancho de banda de la conexión, fundamentalmente.

Se ha aplicado el experimento patrón a ambos laboratorios de instrumentación remota, repitiendo el procedimiento en diferentes momentos del día, midiendo las diferentes métricas descritas anteriormente, con los resultados que se describen a continuación, que constituyen un promedio de las experimentaciones realizadas:

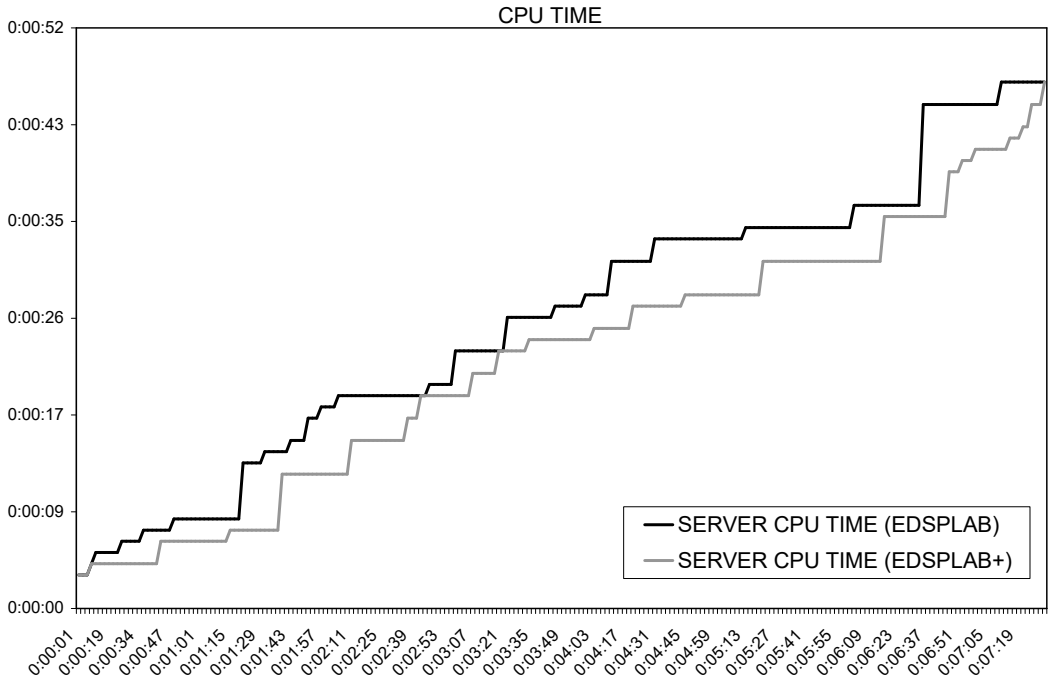


Figura 78.- Tiempo de CPU (CPU Time) en servidor de eDSPlab y eDSPlab+

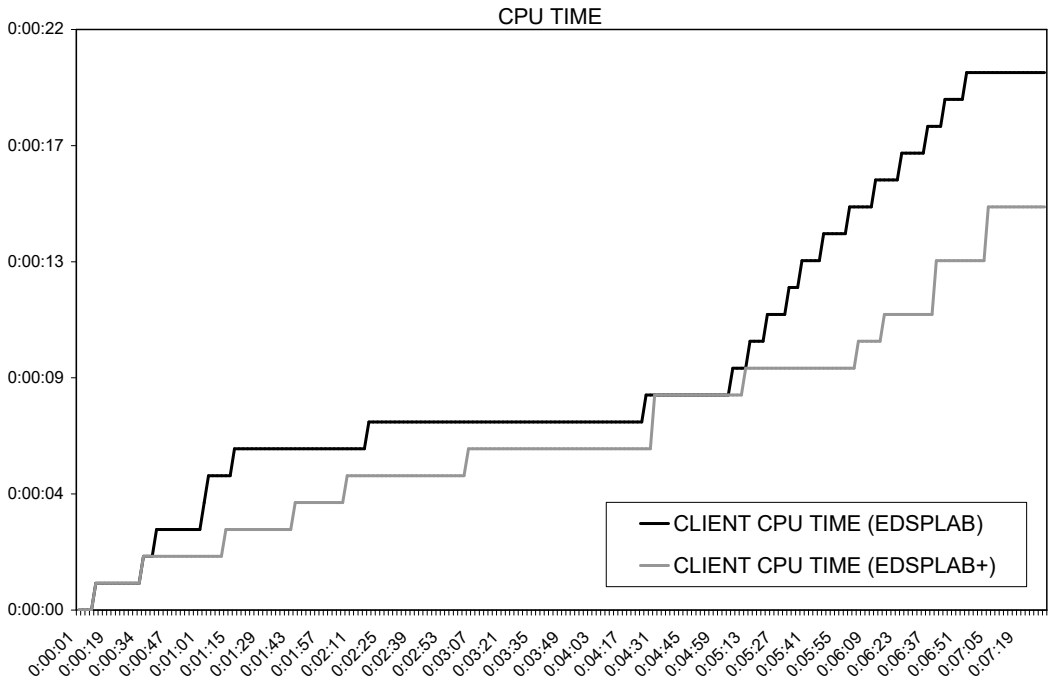


Figura 79.- Tiempo de CPU (CPU Time) en cliente de eDSPlab y eDSPlab+

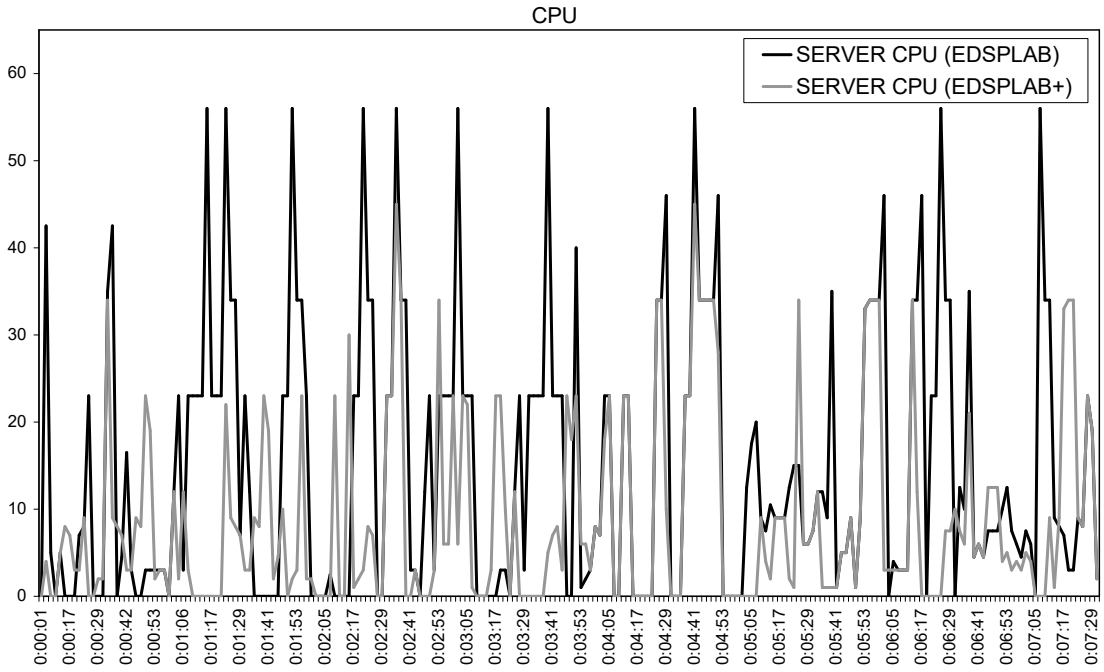


Figura 80.- Uso de CPU (CPU Usage) en servidor de eDSPlab y eDSPlab+

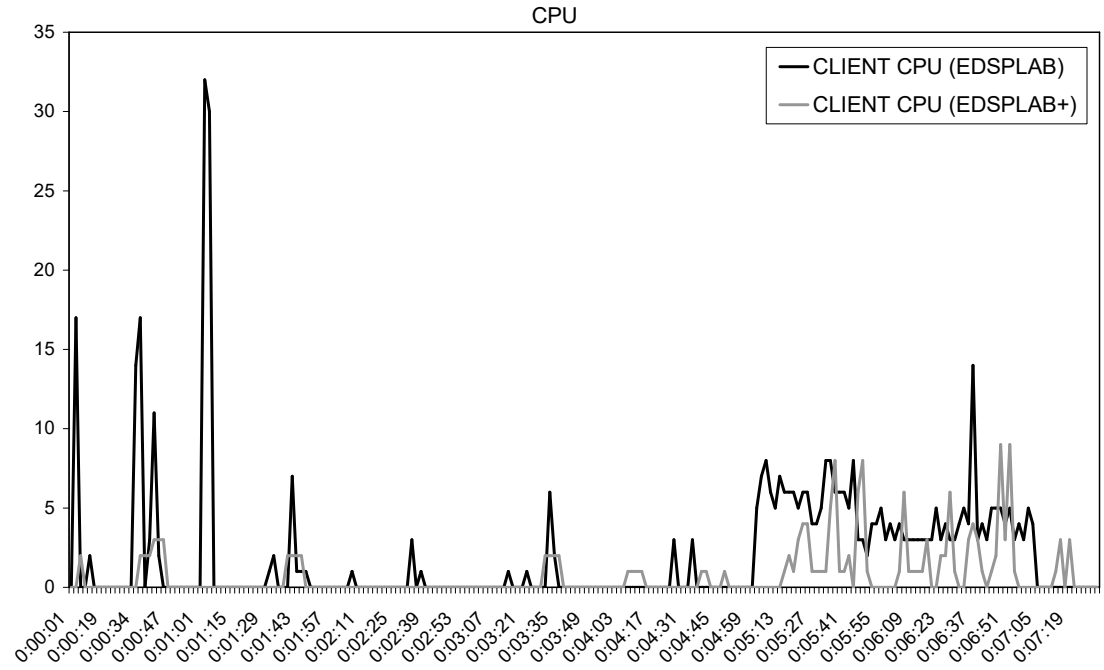


Figura 81.- Uso de CPU (CPU Usage) en cliente de eDSPlab y eDSPlab+

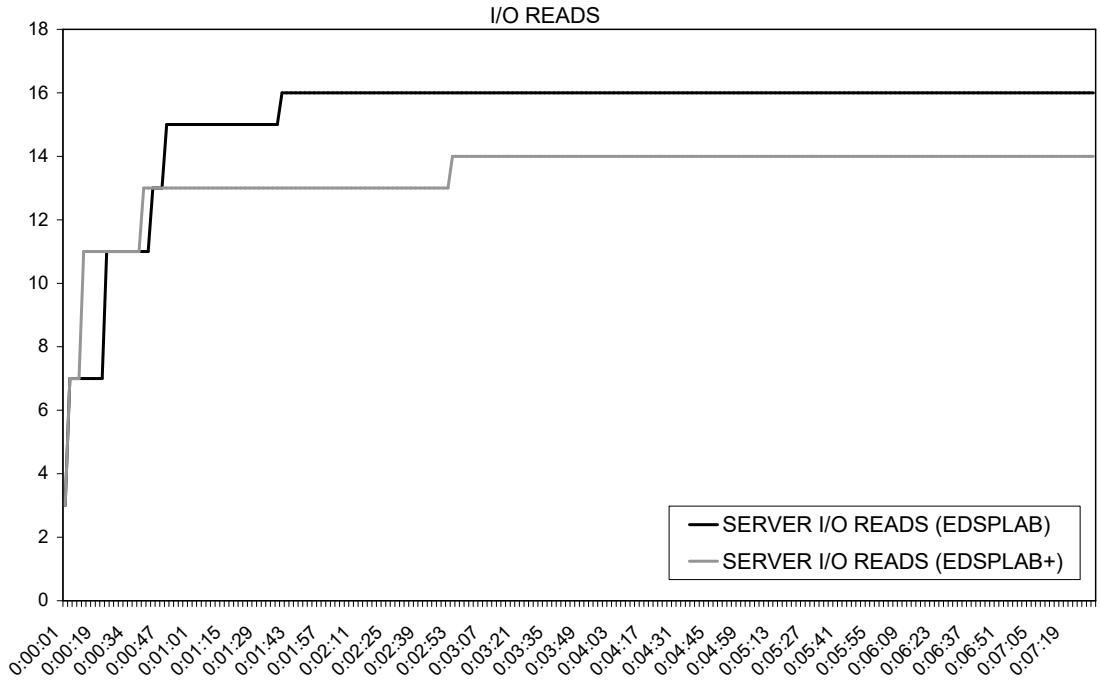


Figura 82.- Lecturas E/S (I/O Reads) en servidor de eDSPlab y eDSPlab+

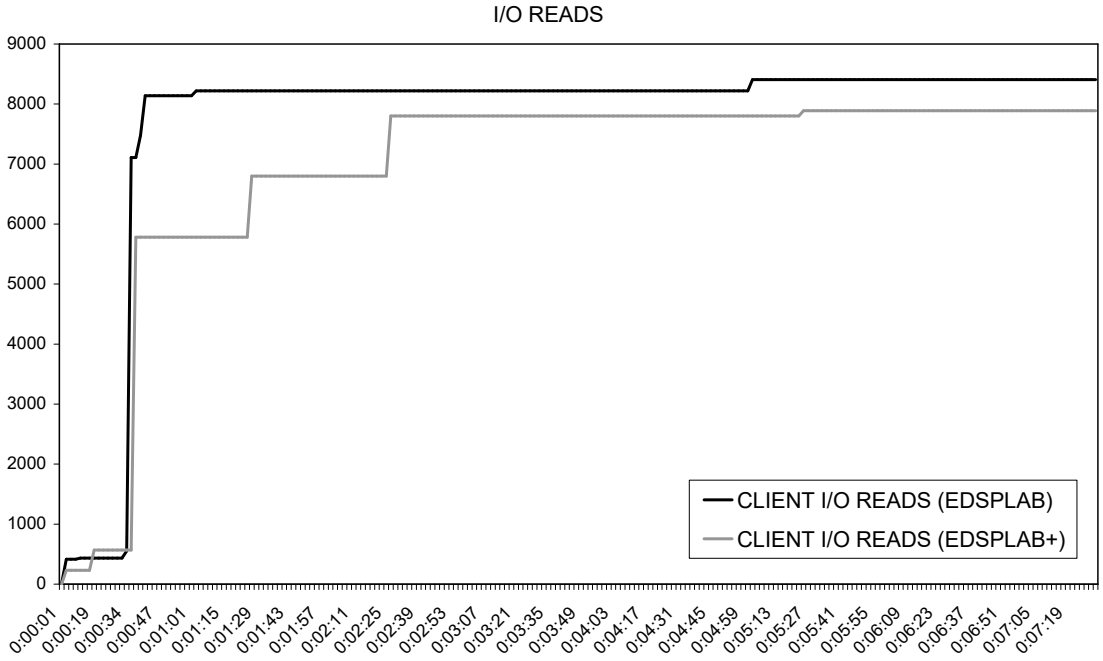


Figura 83.- Lecturas E/S (I/O Reads) en cliente de eDSPlab y eDSPlab+

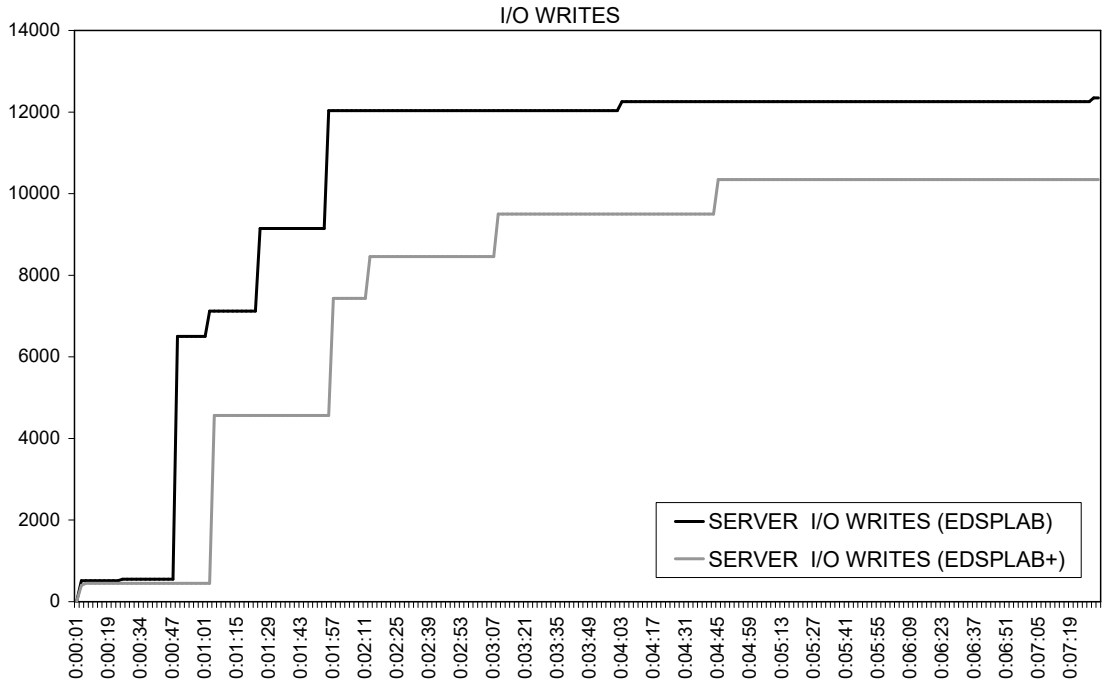


Figura 84.- Escrituras E/S (I/O Writes) en servidor de eDSPlab y eDSPlab+

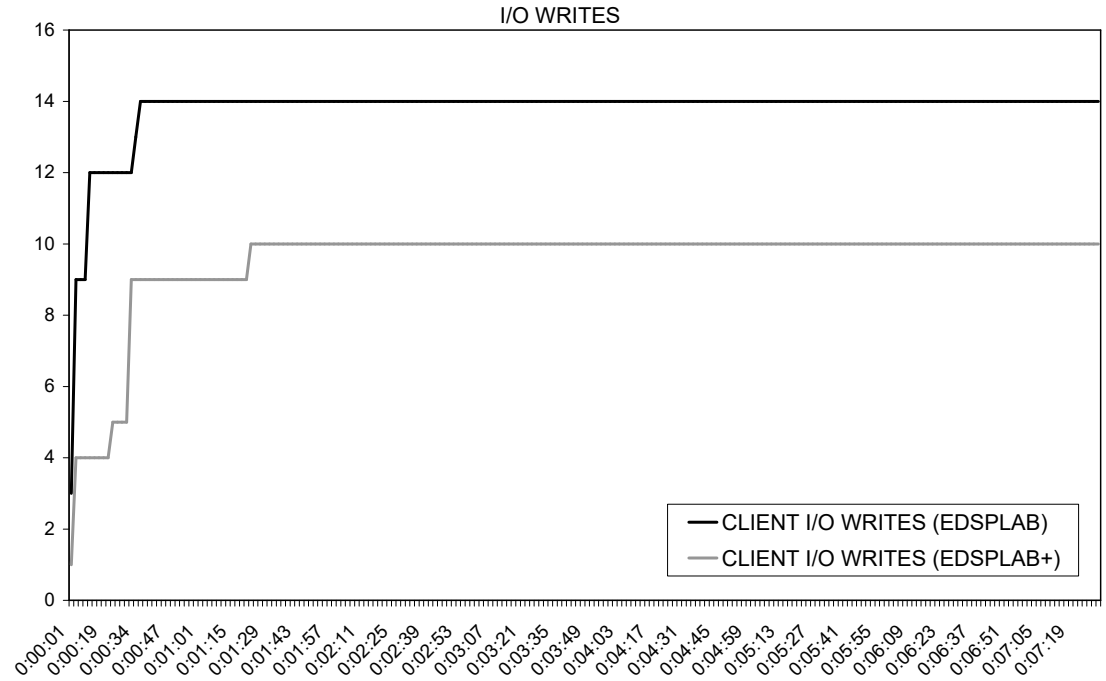


Figura 85.- Escrituras E/S (I/O Writes) en cliente de eDSPlab y eDSPlab+

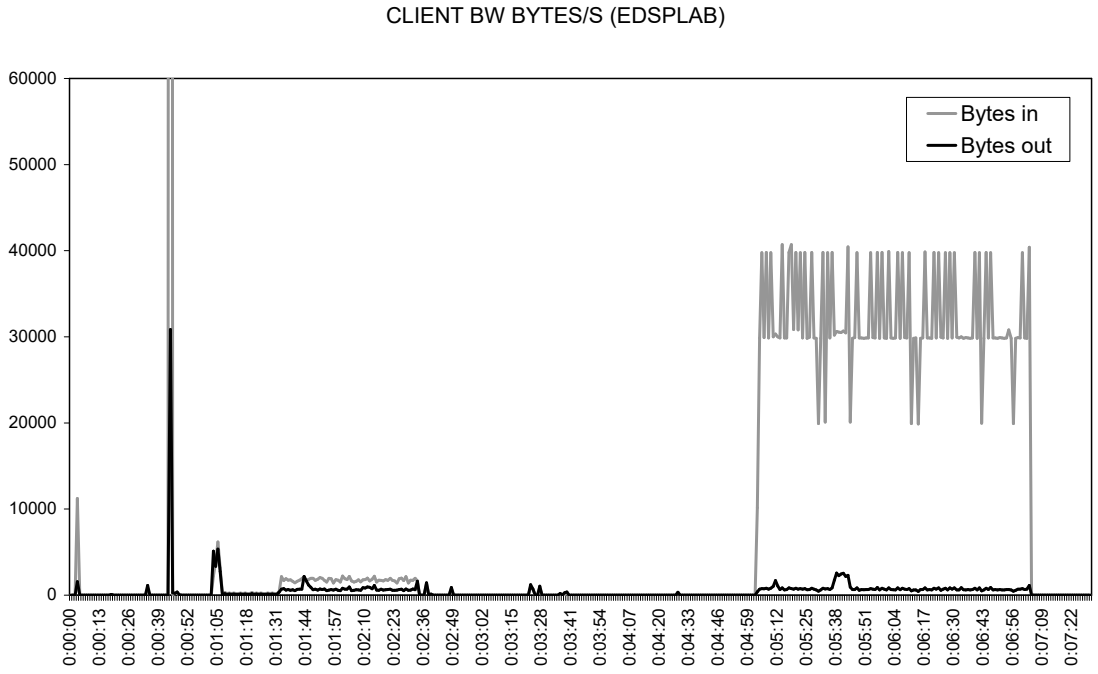


Figura 86.- Ancho de banda ocupado en cliente de eDSPlab

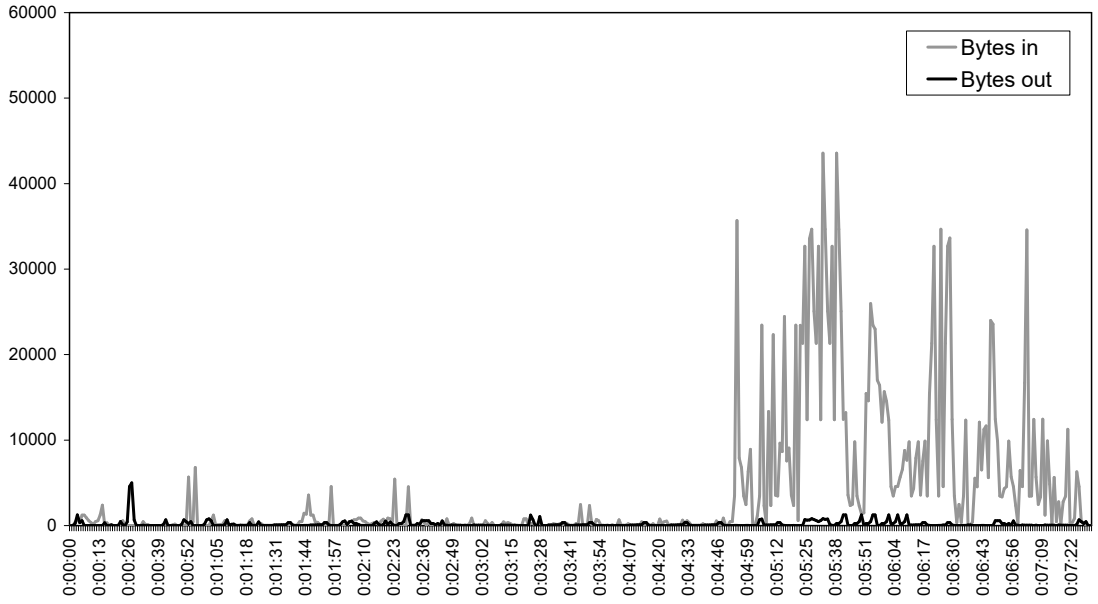


Figura 87.- Ancho de banda ocupado en cliente de eDSPlab+

Las métricas propuestas en la medición de parámetros objetivos arrojan algunos resultados de interés. Efectivamente la utilización de un modelo basado en tecnología AJAX para implementar la interfaz con el cliente libera considerablemente a la CPU del mismo; estimándose una mejora promedio del 26% y el 65% en los parámetros CPU time y CPU usage respecto de la

solución basada en LabVIEW. Por otra parte, pese a que podría pensarse que el empleo de tecnologías basada en JAVA en el extremo servidor podría ralentizar el comportamiento del mismo; la sencillez de la implementación y el reducido número de procesos implicados también se manifiestan en una reducción de los parámetros asociados a la CPU, contando con una mejora del 14% y 41% en los parámetros CPU time y CPU usage en eDSPlab+.

Los resultados anteriormente descritos también son manifiestos en los parámetros relacionados con el número de operaciones de escritura y/o lectura, aunque dicha mejora es menos significativa.

Por último, el ancho de banda ocupado, tanto de entrada como de salida (BW in y BW out), ponen de manifiesto el cumplimiento de otro de los objetivos marcados en el presente trabajo, reducir el ancho de banda ocupado por el sistema de instrumentación; lo que se debe, principalmente, a la utilización de la tecnología de comunicación asíncrona cliente-servidor implementada y el uso de AJAX en el diseño de la interfaz del cliente, lo que implica la actualización de, únicamente, aquellas partes de la interfaz estrictamente necesarias. Así, se observa una mejora del 68% en el ancho de banda de entrada y del 61% en el de salida.

	$\overline{X}_{eDSPlab}$		$\overline{X}_{eDSPlab+}$		$\frac{\overline{X}_{eDSPlab} - \overline{X}_{eDSPlab+}}{\overline{X}_{eDSPlab}}\%$	
	Cliente	Servidor	Cliente	Servidor	Cliente	Servidor
$\overline{X}_{CPU-time}$	0:00:09	0:00:26	0:00:07	0:00:22	25,92	13,94
$\overline{X}_{CPU-usage}$	2,00	14,92	0,71	8,75	64,59	41,36
$\overline{X}_{I/O-writes}$	13,79	10398,57	9,54	7825,64	30,82	24,74
$\overline{X}_{I/O-reads}$	7742,53	15,25	6938,14	13,38	10,39	12,21
$\overline{X}_{BW-in}$	12386,44		3945,41		68,15	
$\overline{X}_{BW-out}$	447,99		176,80		60,54	

Tabla 7.- Métricas objetivas de medición de la QoE

En [161] podemos encontrar una comparativa, empleando los mismos parámetros, del modelo de laboratorio de instrumentación LabVIEW descrito, eDSPlab, basado en la utilización de un cliente grueso frente al modelo de cliente delgado.

6.3. Evaluación subjetiva de la QoE

La evaluación subjetiva se basa en la percepción personal de la calidad del servicio que un sistema ofrece a un usuario. Si bien existen diversas normas que permiten definir procedimientos y escalas de medida para evaluar este parámetro en diversos sectores o aplicaciones, como la telefonía o los sistemas de vídeo; los sistemas de instrumentación remota no disponen de tales referencias.

En el presente trabajo se propone como método de evaluación analizar la aceptación que tiene el uso de los laboratorios de instrumentación remota por parte de un usuario final en comparación con su análogo presencial. Para dicha evaluación, resulta imprescindible enmarcar el sistema de instrumentación dentro de un contexto (educativo, investigación o profesional, entre otros).

Para poder evaluar desde el punto de vista del usuario final tanto el uso de la herramienta como las variables con una influencia relevante en su uso se han empleado los modelos de aceptación tecnológica (TAM), englobados dentro de la teoría de los sistemas de información. Este modelo ha sido aplicado a eDSPlab [162]. Los resultados obtenidos muestran las principales fortalezas y debilidades del sistema propuesto.

La utilización de los modelos TAM se basa en modelar el uso de nuevas tecnologías mediante la utilidad y facilidad de uso; a lo que se suman un conjunto de variables externas con influencia sobre éstas. Para ello partimos de un cuestionario que persigue medir aquel conjunto de dimensiones que son consideradas relevantes en relación a la influencia que pueden tener en el uso de un nuevo sistema o tecnología, en nuestro caso, el uso de eDSPlab [162], [163], [164]. El cuestionario empleado consta de cincuenta y nueve preguntas valoradas empleando una escala Likert de 1 a 7, en la que “1” expresa “fuertemente desacuerdo” y “7” fuertemente de acuerdo.

Dimension (Reliability)		Item (Correlation item-dimension)
Format (0.7554)	I1	The instructional material is presented in a flexible order (0.3953)
	I2	Ease of readability/understandability of the text (0.5766)
	I3	The information provided by the tool is presented in a useful format (0.5668)
	I4	Clarity of the instructional material (0.6949)
Methodology (0.7601)	I5	The scope and the goals of the tool are clearly defined (0.6706)
	I6	The length of the content blocks are appropriate (0.4696)
	I7	The tool is adapted to the teaching content (0.6403)
Feedback (0.7849)	I8	Useful feedback from the tool (0.5135)
	I9	Receipt of timely feedback from the system (0.7431)
	I10	Feedback from the system stimulates the use of the tool (0.5782)
User adaptation (0.7537)	I11	The tool allows supervising the activity of users (0.4689)
	I12	The users have the possibility of choosing to solve the problems they like (0.4515)
	I13	The tool is adapted to different user profiles (0.5543)
	I14	The tool has different levels of complexity (0.4314)
	I15	The tool allows users to develop their own initiatives (0.5543)
	I16	Ability to control the sequence of instruction (0.4860)
	I17	The e-learning system enables you to learn the content you need

		(0.3511)
Communicativeness (0.8763)	I18	The e-learning system makes it easy for you to discuss questions with your teachers (0.6845)
	I19	The e-learning system makes it easy for you to discuss questions with other students (0.8117)
	I20	The e-learning system makes it easy for you to share the learned content with other students (0.7964)
Diffusion (0.8562)	I21	Before using the tool, I know all its features (0.7507)
	I22	I know the technical requirements for using the tool (0.7187)
	I23	I know the formative requirements for using the tool (0.7606)
	I24	Lecturers promote the use of the eLearning tool (0.5673)
Accessibility (0.7980)	I25	Accessibility of the eLearning tool (0.6379)
	I26	The communication channel is appropriate for accessing the tool (0.6319)
	I27	The communication channel for accessing the tool is easy to use (0.6519)
Interactivity and control (0.8244)	I28	Ability to use a variety of methods (menu, button,2) to interact with the system (0.2490)
	I29	Extent to which the system enables the subjects to actively interact with it (0.6742)
	I30	Ability to control the pace of instruction (0.7968)
	I31	Ability to control the sequence of instruction (0.7333)
	I32	Ability to select components or modules of instruction needed to acquire the various concepts (0.7251)
Enjoyment and playfulness (0.8661)	I33	Using the eLearning tool is pleasant (0.6580)
	I34	I have fun using the eLearning tool (0.7010)
	I35	The content of the tool is showed in an amusing way (0.7540)
	I36	The graphic design is attractive to users (0.6085)
	I37	Using the eLearning tool excites my curiosity (0.6922)
	I38	Using the eLearning tool arouses my imagination (0.5579)
Reliability (0.8361)	I39	The tool is robust and works properly (0.6310)
	I40	The tool is reliable (0.6114)
	I41	Whenever I use the tool, it works properly (0.7482)
	I42	I am confident about the security of the tool (0.6789)
User's tool (0.6178)	I43	The tool includes a search tool (0.4228)
	I44	The tool shows the different events of my learning process (0.4731)
	I45	The tool includes forums and chats (0.3691)
Easy to use (0.9081)	I46	I find it easy to get the eLearning tool to do what I want it to do (0.8029)
	I47	My interaction with the eLearning tool is clear and understandable (0.8246)
	I48	I find the eLearning tool easy to use (0.8905)
	I49	Interacting with the eLearning tool will not require a lot of my mental effort (0.6683)
Utility (0.9366)	I50	Using the eLearning tool would improve my performance in this course (0.7983)
	I51	Using the eLearning tool would increase my productivity in this course (0.8915)
	I52	Using the eLearning tool would enhance my effectiveness in this course (0.8243)
	I53	I find the eLearning tool would be useful in this course (0.7731)
	I54	Using the eLearning tool in my job would enable me to accomplish tasks more quickly (0.7705)
	I55	Using the eLearning tool would make it easier to do my task (0.8085)

Use intention (0.8248)	I56	I intend to review some concepts using the eLearning tool frequently (0.7018)
	I57	I intend to compare some theoretical and practical concepts explained in classes with the point of view of the eLearning tool (0.7018)
Use (0.8008)	I58	How many times per week have you used the eLearning tool? (Average value) (0.6678)
	I59	How many hours per week have you used the eLearning tool? (Average value) (0.6678)

Tabla 8.- Cuestionario validado mediante el alfa de Cronbach

Para poder distribuir el cuestionario a una población real de usuarios, el laboratorio eDSPlab ha sido implementado dentro del contexto de una asignatura de la titulación de Ingeniería de Telecomunicación de la Universidad de Sevilla, “Complementos de Sistemas Electrónicos Digitales” y ha sido distribuidos a una población de alumnos para su evaluación.

Las preguntas del cuestionario se agrupan por dimensiones a considerar en el modelo: formato, metodología, realimentación, adaptación al usuario, comunicatividad, difusión, interactividad y control, amigabilidad, fiabilidad, herramientas de usuario, facilidad de uso, utilidad, intención de uso y uso. Entre todas, las últimas cuatro se corresponden con las dimensiones reflejadas en el modelo TAM original propuesto por [163], mientras que las otras dimensiones son las variables externas con influencia en este modelo base.

Con objeto de validar el presente cuestionario se ha utilizado el alfa de Cronbach, un índice frecuentemente empleado para determinar la fiabilidad con la que las distintas cuestiones o ítems que constituyen una dimensión miden la variable subyacente. Este coeficiente puede variar entre 0 y 1; estableciéndose un límite inferior mínimo de 0,7 [165]. En la Tabla 8 se muestra el valor del índice de Cronbach calculado entre paréntesis debajo de cada dimensión.

Los datos obtenidos son procesados empleando los Modelos de Ecuaciones Estructurales [166], una técnica multivariante que combina aspectos de la regresión múltiple, examinando relaciones de dependencia y análisis factorial, que representa conceptos inmedibles – factores – con variables múltiples, a objeto de poder estimar un conjunto de relaciones de dependencia interrelacionadas de forma simultánea. Entre los posibles modelos existentes, se empleará PLS para obtener el modelo de la Figura 88 para eDSPlab.

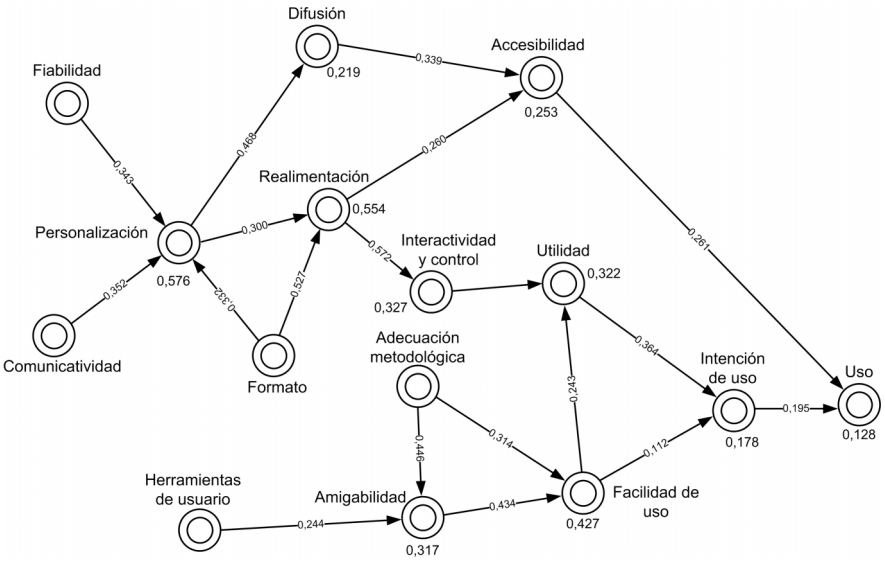


Figura 88.- Modelo de aceptación tecnológica de eDSPlab

Observando la Figura 88 podemos destacar la influencia de la accesibilidad sobre el uso, lo cual resulta intuitivamente evidente, pues una herramienta remota debe ser, ante todo, accesible, pues es uno de los objetivos que se persiguen cuando se ponen a disposición de la comunidad de usuarios. Gracias a la utilización de interfaces accesibles a través de Internet y la posibilidad de emplear el sistema de instrumentación a cualquier hora, la percepción subjetiva del usuario en relación a la accesibilidad se ve incrementada. Existen, a su vez, otras variables, como la difusión, que, a través de la accesibilidad, influyen en parte en el uso.

La facilidad de uso también se ve influenciada por la amigabilidad y adecuación metodológica, aunque esta última tenga menos influencia que la primera, de lo que se deduce la necesidad de diseñar una interfaz más amigable, aspecto que se persigue en eDSPlab+.

La utilidad es la que contribuye fundamentalmente a la intención de uso y, a su vez, observamos que surge una influencia que también resulta intuitiva desde el punto de vista de la percepción subjetiva del usuario final: la interactividad y el control. En el diseño de los sistemas de instrumentación remotos, este es uno de los aspectos a tener en consideración más importantes, pues la funcionalidad y control que ofrece la interfaz remota debe ofrecer la mayor semejanza posible con el sistema de instrumentación presencial; de tal manera que el usuario remoto pueda interactuar con el experimento (DUT) e intercambiar información con la interfaz “en tiempo real”; percibiéndolo del mismo modo que si estuviera físicamente en el laboratorio.

Otra variable que se manifiesta como elemento influyente en el sistema es la realimentación, lo que implica que el laboratorio de instrumentación remota debe proveer al usuario final de información precisa que le permita determinar si está operando correctamente o no sobre el sistema; lo que le sugiere la sensación subjetiva de “control” sobre lo que está haciendo. En este sentido, las opciones que ofrece la tecnología empleada para la implementación de eDSPlab están muy limitadas; sin embargo, el uso de tecnologías propiamente orientadas al manejo y presentación de la

información, como son las empleadas en eDSPlab+ facilita enormemente esta labor, existiendo infinidad de posibilidades de mejorar esta dimensión.

Las variables asociadas a la fiabilidad, comunicatividad o el formato afectan a la personalización; aspecto influyente, especialmente, en la difusión.

En resumen, desde el punto de vista del usuario final, eDSPlab presenta un aceptable grado de accesibilidad, manifestándose una deseable una mejora en los mecanismos de control e información en tiempo real; pese a lo cual le resulta un entorno amigable y fácil de usar.

De los anteriores resultados se arroja que la utilización del laboratorio de instrumentación remota presenta una aceptación favorable a la población de estudiantes que lo ha utilizado. Se manifiesta una clara influencia en la facilidad de uso de la dimensión asociada a la amigabilidad de la plataforma, aspecto que se ha desarrollado en eDSPlab+ empleando tecnologías Web que permiten aumentar las prestaciones en cuanto a la ligereza del cliente software necesario para controlar la aplicación, la reducción del tiempo de respuesta frente a los eventos solicitados por el usuario final y la mejora del diseño y apariencia más familiar.

Por otra parte, también se refleja la necesidad de mejora en lo que a la realimentación se refiere, pues el usuario de la herramienta remota puede percibir subjetivamente una falta de control en la experimentación respecto de su análogo presencial. Las tecnologías Web permiten dotar a la interfaz cliente de infinidad de mecanismos que mejoran este aspecto. eDSPlab+ incorpora paneles de realimentación que permiten conocer exactamente el estado y proceso del entorno del DSK, la disponibilidad de controles de ayuda en línea, así como una infinidad de posibilidades relacionadas con la incorporación de otras tecnologías como aplicaciones multimedia [129] [131], integración dentro de entornos LMS como Moodle o, incluso, la integración de entornos de realidad virtual que permiten visualizar la interfaz del sistema en 3D [134], [135].

7. Conclusiones y trabajos futuros

El estudio del estado del arte y la técnica de los sistemas de instrumentación remota arroja que, desde años atrás, ha existido un gran interés en el desarrollo de este tipo de plataformas. Encontramos numerosas propuestas basadas en potentes paquetes de desarrollo comerciales que las firmas ponen a nuestra disposición, las cuales permiten un desarrollo rápido y sencillo, con multitud de posibilidades y opciones. Por otra parte, dado el carácter multidisciplinar de los sistemas de instrumentación remota, comienzan a desarrollarse nuevas propuestas y modelos que persiguen mejorar la facilidad de uso, la flexibilidad, el rendimiento y funcionalidad de estas arquitecturas. En este sentido aparecen necesidades constructivas como son la modularidad y la portabilidad, dos dimensiones que facilitan el ciclo de vida de las soluciones de instrumentación, al tiempo que aprovechar aquellos bloques reutilizables, reduciendo el coste de desarrollo.

El presente trabajo de tesis ha pretendido desarrollar dos modelos de laboratorio de instrumentación remota basados en estas dos tendencias: Una asociada a las plataformas comerciales de desarrollo de aplicaciones de instrumentación remota; centrándonos en el paquete más utilizado actualmente, LabVIEW. Y una segunda propuesta que aprovecha la potencialidad de las nuevas tecnologías que acontecen a nuestro alrededor; especialmente las que a los servicios Web se refieren, presentando una arquitectura de sistema o laboratorio de instrumentación remota basado en tecnologías de entornos Web. En este sentido, cabe destacar que el uso de tecnologías como AJAX junto a los Servlets de Java y el Java Native Interface, fundamentalmente, han permitido desarrollar un nuevo modelo que mejora algunos aspectos importantes que afectan a cómo el usuario final percibe la herramienta.

El primer desarrollo, eDSPlab, nos ha permitido crear una arquitectura que, comenzando por un acceso local a un sistema únicamente de instrumentación, ha convergido a un entorno “integrado” en una plataforma Web accedida a través de Internet. Muchas son las virtudes arrojadas de este sistema; como la rapidez de diseño, el escaso conocimiento de programación necesario para llegar a soluciones finales, lo que hace muy sencillo su uso, al tiempo que las firmas comerciales auspician estas plataformas con complementos y paquetes adicionales que dan solución y soporte a las demandas de la comunidad de usuarios de la misma. No obstante, el desarrollo el laboratorio y su posterior evaluación desde el punto de vista del usuario final reflejaron la necesidad de algunas mejoras, algunas de ellas difícilmente alcanzables con el entorno comercial base. La integración con plataformas CMS o LMS también presenta dificultades y el modelo de comunicación cliente-servidor síncrono, basado en la utilización de un plug-in en el extremo cliente también supone algunos inconvenientes; algunos acotables con módulos adicionales que ofrece la empresa proveedora, en muchos casos, inalcanzable por muchas empresas, organismos y entidades dado el elevado coste de las mismas.

La necesidad de mejora de eDSPlab supuso el planteamiento inicial de la nueva arquitectura, eDSPlab+, donde se pone de manifiesto la posibilidad de

desarrollar plataformas de instrumentación remota con un reducido coste económico. La utilización de tecnologías multiplataforma orientada a objetos, como es Java, nos brinda una elevada capacidad de portabilidad y modularidad, el acceso al “escritorio” del servidor, es decir, al hardware del mismo, necesario para acceder a la tarjeta de control de la instrumentación, se resuelve con una tecnología muy útil pero poco conocida entre los programadores, el Java Native Interface. Los distintos problemas que se han ido presentando han sido explicados a lo largo del documento y se les ha dado solución; caso de la problemática de los distintos comandos entre equipos del mismo tipo, por ejemplo, lo que se ha solucionado empleando unos archivos asociados a cada dispositivo y aislando la petición del cliente al servidor del envío de comandos del controlador a la instrumentación. La necesidad de contemplar la existencia de multitud de nuevos dispositivos con necesidad de conexión al sistema supone perseguir dos objetivos, uno asociado a la utilización de tecnologías totalmente compatibles en el extremo cliente, con un modelo de cliente delgado que no libere de la necesidad de descargar aplicaciones que se ejecuten sobre su máquina; lo que resulta imposible en los terminales de algunas empresas o instituciones; y por otro lado aislar la tecnología de acceso del cliente de la tecnología de gestión de peticiones en el extremo servidor. La solución aportada se basa en la utilización de búferes que actúan como interfaz de las peticiones y el uso de tecnologías Web potencialmente compatibles con la mayoría de los navegadores. Si bien es cierto que tecnologías como Flash darían mayor interactividad a la interfaz gráfica, su uso se está abandonando debido a los problemas de compatibilidad, precisamente, con dispositivos móviles. Una tecnología que mejora este problema es AJAX, basado en el uso de HTML y JavaScript fundamentalmente; dos lenguajes que la mayoría de dispositivos soportan. En el documento se describe el conjunto de tecnologías que han puesto de manifiesto la posibilidad de desarrollar una interfaz ligera para el cliente y eficiente, al emplear el modelo de comunicación cliente-servidor asíncrono; en contraposición con el modelo de eDSPlab, al mismo tiempo que únicamente se actualiza aquellas partes de la interfaz estrictamente necesarias; reduciendo el ancho de banda utilizado y mejorando la rapidez de la misma, lo cual percibe como una bondad el usuario final.

Finalmente, se han propuesto un conjunto de métricas para la evaluación de la calidad que ofrece este tipo de sistemas, obteniendo como resultado una mejora desde el punto de vista técnico de la implementación basada en tecnologías Web. Pese a que se esperaba un empeoramiento de la nueva propuesta en relación al uso de los recursos del servidor, dado que la ejecución de Java es descrita por muchos autores como poco eficiente, al precisar de la máquina virtual Java para su ejecución; los resultados arrojan una mejora tanto en el uso de los recursos hardware del cliente como del servidor y una reducción del ancho de banda. Esto se explica de la reducción de procesos necesarios para dar respuesta a las peticiones del cliente, al crear canales de comunicación con aquellas partes de la interfaz cliente que precisan actualizarse únicamente.

Si bien se ha analizado la aceptación tecnológica de eDSPlab dentro de un contexto educativo; resultaría conveniente aplicar el nuevo diseño a una población de similares características y comparar los resultados obtenidos, del

mismo modo que se ha procedido con las métricas objetivas; sin embargo dicho estudio queda fuera del ámbito del presente trabajo de tesis y se propone como futura línea de investigación; resultados que pueden aportar, más allá de cuantas mejoras o innovaciones técnicas podamos proponer, que dimensiones son mejorables y tienen aportación desde el punto de vista de la percepción subjetiva que tiene el usuario final del uso.

En este sentido, algunas de las líneas de mejora y futuros trabajos que se proponen a tenor del trabajo expuesto en la presente tesis son los que se describen a continuación.

Los laboratorios de instrumentación propuestos se han centrado en una interfaz de instrumentación GPIB, por lo que se propone el estudio de aplicación del modelo propuesto en eDSPlab+ empleando instrumentación LNA y LXI y utilizando librerías IVI y VISA.

Otra línea de trabajo, dentro del concepto propuesto en eDSPlab+, es dotar al sistema de mecanismos de creación y descubrimiento que permitan reconocer la incorporación de nuevos equipos al sistema, actualizándose automáticamente la interfaz Web. Este tipo de mecanismos ya está presente en en diversas tecnologías y sectores, caso del bus industrial AS-i o los sistemas de automatización de edificios [138], [139] y resulta posible acometerlo gracias al diseño modular propuesto.

Otras dos mejoras contempladas inicialmente en la solución eDSPab+ son el estudio de la posibilidad de aislar la dependencia con el sistema operativo del DSK y la utilización de la tecnología “drag&drop” para la configuración de consolas de equipos de forma personalizada.

Para favorecer el uso multiusuario del laboratorio de instrumentación remota, resultaría conveniente estudiar la posibilidad de establecer un módulo que permita la multiplexación de la herramienta entre un grupo reducido de usuarios en tiempo real así como establecer mecanismos tipo “Token” para la gestión de las reservas y tiempo de uso, detectando, entre otros factores, la inactividad durante un extenso periodo de tiempo; esto contribuiría a mejorar la dimensión asociada a la disponibilidad de los laboratorios.

Si bien el uso de la tecnología AJAX en el extremo cliente permite su ejecución en dispositivos móviles; se propone la realización de un profundo estudio del estado actual y futuro de la técnica en tecnologías de acceso Web móviles y la adaptación de la interfaz actualmente desarrollada a dichos sistemas.

El uso de la tecnología desarrollada no tiene aplicación exclusivamente en el ámbito de la instrumentación remota; sino todo lo contrario. Uno de los sectores donde este tipo de tecnologías comienza a resultar atractiva es el la domótica e inmótica; especialmente lo que a las interfaces de los usuarios de hogares digitales y edificios inteligentes se refiere, pudiendo desarrollarse nuevos modelos utilizando el modelo propuesto. Sectores como el Internet de las Cosas o la instrumentación ubicua ya se encuentran en los laboratorios de I+D+i de empresas de base tecnológica y comerciales del sector, trabajando sobre estas tecnologías para dar soluciones de hogar digital.

8. Referencias

[1] P. Riu, J. Ramos, J. Rosell, *Sistemas de instrumentación*", Cataluña: Ediciones UPC, 2000.

[2] H. A. Mendiburu, *Instrumentación Virtual Industrial,* Perú: INDECOPI, 2006.

[3] R. M. Mujal, *Tecnología eléctrica,* Cataluña: Edicions UPC, 2006.

[4] P. Allen, *Física preuniversitaria,* Reverte, 1991.

[5] José Ramón Bertomeu Sánchez, Antonio García Belmar. *Abriendo las cajas negras: colección de instrumentos científicos de la Universitat de València*". Valencia: Edición Universitat de València, 2002.

[6] Robert Bud, Deborah Jean Warner. *Instruments of science: an historical encyclopedia.* Taylor & Francis, 1998.

[7] "The morals of Energy MeTering: Constructing and Deconstructing the Precision of the Victorian Electrical Engineer's Ammeter and Voltmeter". En: N. WISE (ed.), *The Values of Precision*, Princenton, Univ. Press, 239-283.

[8] V. Philips, *Waveforms: A History of Early Oscillography*, Bristol, Hilger. 1997.

[9] June Jamrichoja Parsons, Carl McDaniel. *Conceptos de computación: nuevas perspectivas*, Cengage Learning Editores, 2008.

[10] José Rafael Lajara Vizcaíno, José Pelegrí Sebastiá. Colaborador José Pelegrí Sebastiá. *LabVIEW: Entorno gráfico de programación*, Marcombo, 2007.

[11] "Evolución de la instrumentación electrónica programable", Heriberto I. Hernández Martínez, Ángel F. González Hernández y Josué N. García Matías, Instituto de Electrónica y Computación, Universidad Tecnológica de la Mixteca, TEMAS DE CIENCIA Y TECNOLOGÍA vol.10 número 30 septiembre-diciembre 2006, pp 33 – 42

[12] Mànuel Làzaro, Antoni Biel Solé, Domingo Olivé Duran, Joaquim Prat Tasias, Jordi Sánchez Robert, Francesc Joseph, *Instrumentación virtual. Adquisición, procesado y análisis de la señal.* Ediciones UPC, 2001.

[13] Helsel, R. 1998, "Visual Programming with HP VEE", Hewlett Packard Professional Books, third edition, 1998.

[14] Walt Boyes, Butterworth-Heinemann. *Instrumentation reference book.* 2003.

[15] A. Lazaro, *LabVIEW 6i, Programación Gráfica para el Control de Instrumentación*. Paraninfo-Thomson Learning. 2001.

[16] "Modelo de referencia de laboratorios virtuales y aplicación a sistemas de teleeducación", Vicent Miquel Rodrigo Peñarrocha, 2003.

[17] http://digital.ni.com/worldwide/latam.nsf/web/all/01E4BFF8EC93532086256B6000669953

[18] Rak, R.J.; Michalski, A., “Education in Instrumentation and Measurement: The Information and Communication Technology Trends”, IEEE Instrumentation & Measurement Magazine, Volumen 8, Número 2, pp. 61 – 69, Marzo 2005.

[19] Goldberg, H., “What is Virtual Instrumentation”, IEEE Instrumentation & Measurement Magazine, Volumen 3, Número 4, pp. 10-13, Diciembre 2000.

[20] Prasad Calyam, Abdul Kalash, Ramya Gopalan, Sowmya Gopalan, and Ashok Krishnamurthy , “RICE: A Reliable and Efficient Remote Instrumentation Collaboration Environment”, Advances in Multimedia, Volumen 2008, 2008.

[21] M.J. Callaghan_, J. Harkin, E. McColgan, T.M. McGinnity, L.P. Maguire , “Client–server architecture for collaborative remote experimentation”, Journal of Network and Computer Applications, Volumen 30, Número 4, pp. 1295–1308, Noviembre 2007.

[22] Chirico, M., Scapolla, A.M., Bagnasco, A., “A new and open model to share laboratories on the Internet”, IEEE Transactions on Instrumentation and Measurement, Volumen 54, Número 3, pp. 1111-1117, Junio 2005.

[23] Chindris, G., Hedesiu, H., “Virtual instrumentation platform for power electronics education”, ISSE, 32nd International Spring Seminar on Electronics Technology, pp. 1-4, Mayo 2009.

[24] Ping-Huang Wu, Chin-Hwa Kuo, Ching-Chung Lin, “Design and implementation of the remote control lab using PDA”, . WMTE, IEEE International Workshop on Wireless and Mobile Technologies in Education, pp. 70-72, Noviembre 2005. 2005.

[25] Hsiung Cheng Lin, “Development of Monitoring and Control Precise Riveting System using Wireless Internet”, IEEE 32nd Annual Conference on Industrial Electronics, IECON 2006, pp. 4420-4424, 6-10 Nov. 2006.

[26] Francesco Adamo, Filippo Attivissimo, Giuseppe Cavone, and Nicola Giaquinto, “SCADA/HMI Systems in Advanced Educational Courses”, IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 1, pp. 4-10, 2007.

[27] www.ringrid.eu

[28] The RINGrid project deliverable D4.3 “Final report”

http://www.ringrid.eu/public/deliverables/RINGRID-WP4-D4_3-2008-02-13-Final.pdf

[29] http://zone.ni.com/devzone/cda/tut/p/id/5935

[30] Artículo: Benetazzo, L.; Bertocco, M.; Narduzzi, C.; “Networking automatic test equipment environments”, IEEE Instrumentation & Measurement Magazine Volumen 8, Número 1, pp. 16-21, Marzo 2005.

[31] *Standard Commands for Programmable Instruments*-SCPI. 1999. http://www.ivifoundation.org/specifications/default.aspx

[32] http://www.ni.com/dataacquisition/esa/whatis.htm

[33] http://www.ni.com/dataacquisition/esa/

[34] National Instruments: The Measurement and Automation Catalog. National Instruments Co., 2005.

[35] “Los osciloscopios compatibles con LXI refuerzan la eficiencia de los sistemas ATE”, 13 de octubre de 2007, Jae-yong Chang, Agilent Technologies, Revista Española de Electrónica.

[36] LXI: http://www.lxistandard.org/

[37] Drenkow, G. , “Future test systems architectures”, IEEE Instrumentation and Measurement Magazine, Vol. 8, Issue 3, August 2005.

[38] B. A. Coghlan and M. G. Taylor, "Digitiser/processor for extraction of clinical parameters from Doppler-shift waveforms”, Medical and Biological Engineering and Computing, Volumen 18, Número 1, Enero de 1980

[39] Gomez, H.A.G.; Spinelli, J.C., ”High-linearity, high-immunity data acquisition system for laboratory use”, Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Volumen 3, pp. 1222, 4-7 Noviembre de 1988

[40] Zheng, C., Low, K.-S., “A Virtual Instrument for the Testing and Performance Evaluation of a Microsatellite Power Supply System”, IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 8, pp. 1808-1815, Agosto 2008.

[41] Pohl, A.A.P., Michelon, G.A., Vidal, S.M., Filho, H.S., Fonseca, K.V.O., “Remote RF and Baseband Video Measurement Laboratory Based Upon Open-Code Software”, IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 3, pp. 556-564, Marzo 2008.

[42] Zulkifli, M.Z., Harun, S.W., Thambiratnam, K., Ahmad, H., “Self-Calibrating Automated Characterization System for Depressed Cladding EDFA Applications Using LabVIEW Software With GPIB”, IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 11, pp. 2677-2681, Noviembre 2008

[43] Bilski, P., Winiecki, W., “A Low-Cost Real-Time Virtual Spectrum Analyzer”, IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 6, pp. 2169-2174, Diciembre 2007.

[44] Rajesh Kumar, B., Sridharan, K., Srinivasan, K.,"The design and development of a web-based data acquisition system", IEEE Transactions on Instrumentation and Measurement, Volumen 51, Número 3, pp. 427-432, Junio 2002.

[45] Antii Sivula, "VXI solutions for avionics testing", Aircraft Engineering and Aerospace Technology, Volumen 70, Número 1, pp. 29-31, 1998.

[46] Pizzica, S., "Open systems architecture solutions for military avionics testing", IEEE Aerospace and Electronic Systems Magazine, Volumen 16, Número 8, pp. 4-9, Agosto 2001.

[47] Beche, J.-F., Bucher, J.J., Fabris, L., Riot, V.J., Shuh, D.K., "High-speed nuclear quality pulse height analyzer for synchrotron-based applications", IEEE Transactions on Nuclear Science, Volumen 49, Número 3, Parte 2, pp. 1195-1198, Junio 2002

[48] Andy Toth, "VXI, PXI, or GPIB: Which to use?", Keithley Instruments, Inc., Abril 2004.

[49] Nikitin, P.V., Rao, K.V.S., "LabVIEW-Based UHF RFID Tag Test and Measurement System", IEEE Transactions on Industrial Electronics, Volumen 56, Número 7, pp. 2374-2381, Julio 2009.

[50] Palani, A., Jayashankar, V., "Virtual Instrument for Lightning Impulse Tests", IEEE Transactions on Power Delivery, Volumen 22, Número 3, pp. 1309-1317, Julio 2007.

[51] Baptista, F. G., Filho, J. V., "A New Impedance Measurement System for PZT-Based Structural Health Monitoring", IEEE Transactions on Instrumentation and Measurement, Aceptado para futura publicación, 2009.

[52] Costas-Perez, L., Lago, D., Farina, J., Rodriguez-Andina, J.J., "Optimization of an Industrial Sensor and Data Acquisition Laboratory Through Time Sharing and Remote Access", IEEE Transactions on Industrial Electronics, Volumen 55, Número 6, pp. 2397-2404, Junio 2008.

[53] Pohl, A.A.P., Michelon, G.A., Vidal, S.M., Filho, H.S., Fonseca, K.V.O., "Remote RF and Baseband Video Measurement Laboratory Based Upon Open-Code Software", IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 3, pp. 556-564, Marzo 2008.

[54] "LXI: GOing Beyond GPIB, PXI and VXI", Nota de aplicación 1465-20, Agilent Technologies, 2006.

[55] "10 Good Reasons to Switch to LXI", Nota de aplicación 1465-21, Agilent Technologies, 2006

[56] Cimino, C., Keithley Instrum. Inc., Cleveland, OH; "Rack-and-stack instruments: often a better choice than VXI or PXI systems", Computing & Control Engineering Journal, Vol. 16, issue 1, febrero-marzo 2005

[57] KEITHLEY. *Understanding New Developments in Data Acquisition, Measurement, and Control - A Practical Guide to High Performance Test and Measurement*. 1st Edition, USA, 2007.

[58] "Test-System Development Guide. Choosing Your Test-System Hardware Architecture and Instrumentation", Nota de aplicación 1465-5, Agilent Technologies, 2004.

[59] M Tooley. *Pc Based Instrumentation And Control- 3rd Edition.* Editorial Elsevier, 2005

[60] Franco Pavese, Alistair B. Forbes. *Data Modeling for Metrology and Testing in Measurement Science.* Springer, 2008

[61] www.ni.com/dataacquisition/companion_software.htm#driver

[62] K.H. Kim, T.G. Lee, S. Baek, S.I. Lee, Y. Chu, Y.O. Kim, J.S. Kim, M.K. Park, Y.K. Oh, "Software development of the KSTAR Tokamak Monitoring System", Fusion Engineering and Design, Volumen 83, Números 2-3, pp. 291-294, Abril 2008.

[63] Seong-Heon Seo, T.G. Lee, "Integration of PXI and VXI digitizers into the Hanbit data acquisition system based on MDSplus", Fusion Engineering and Design, Volumen 81, Números 15-17, pp. 1785-1788, Julio 2006.

[64] W.H. Li, B. Liu, P.B. Kosasih, X.Z. Zhang, "A 2-DOF MR actuator joystick for virtual reality applications", Sensors and Actuators A: Physical, Volumen 137, Número 2, pp. 308-320, Julio 2007.

[65] MCDAQ: www.mccdaq.com/daq-software/tracerdaq-pro.aspx

[66] Louise Hecker, Luda Khait, Desmond Radnoti, Ravi Birla, "Novel bench-top perfusion system improves functional performance of bioengineered heart muscle", Journal of Bioscience and Bioengineering, Volumen 107, Número 2, pp. 183-190, Febrero 2009.

[67] Chang-Kyu Kim, Cheol-Su Kim, Umberto Sansone, Paul Martin, "Development and application of an on-line sequential injection system for the separation of Pu, ^{210}Po and ^{210}Pb from environmental samples", Applied Radiation and Isotopes, Volumen 66, Número 2, pp. 223-230, Febrero 2008.

[68] WINDAQ: http://www.dataq.com/products/software/index.htm

[69] Christopher Dromey, Shawn Nissen, Petrea Nohr, Samuel G. Fletcher, "Measuring tongue movements during speech: Adaptation of a magnetic jaw-tracking system", Speech Communication, Volumen 48, Número 5, pp.463-473, Mayo 2006.

[70] E. Lambooij, M. Pilarczyk, H. Bialowas, J.G.M. van den Boogaart, J.W. van de Vis , "Electrical and percussive stunning of the common carp (Cyprinus carpio L.): Neurological and behavioural assessment", Aquacultural Engineering, Volumen 37, Número 2, pp. 171-179, Septiembre 2007.

[71] http://zone.ni.com/devzone/cda/tut/p/id/3590

[72] http://www.ni.com/lwcvi/esa/

[73] Ján Saliga, Linus Michaeli , “Software for metrological characterisation of PC sound cards”, Computer Standards & Interfaces, Volumen 25, Número 1, pp. 45-55, Marzo 2003.

[74] John Z. Shi, Fengshou Gu, Peter Goulding, Andrew Ball, “Integration of multiple platforms for real-time remote model-based condition monitoring”, Computers in Industry, Volumen 58, Número 6, pp. 531-538, Agosto 2007.

[75] Josko, A.; Rak, R.J.;, “Effective simulation of signals for testing ECG analyzer”, IEEE Transactions on Instrumentation and Measurement, Volumen 54, Número 3, pp. 1019-1024, Junio 2005.

[76] http://www.mathworks.es/store/productIndexLink.do

[77] Nemeth, J.G., Vargha, B., Kollar, I., “Online frequency domain system identification based on a virtual instrument”, Transactions on Instrumentation and Measurement, Volumen 49, Número 6, pp. 1260-1263, Diciembre 2000.

[78] Cataliotti, A., Cosentino, V., Nuccio, S., “A Virtual Instrument for the Measurement of IEEE Std. 1459-2000 Power Quantities”, IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 1, pp. 85-94, Enero 2008.

[79] Cheok, A.D., Zhongfang Wang, “DSP-Based Automated Error-Reducing Flux-Linkage-Measurement Method for Switched Reluctance Motors”, IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 6, pp. 2245-2253, Diciembre 2007.

[80] Serie Schaum, Luis Joyanes Aguilar y Lucas Sánchez García. *Programación en C++. Un enfoque práctico*. Universidad Pontificia de Salamanca. Editorial Mcgraw-hill, 2006.

[81] Logan, W., “Life after Visual Basic 6.0 — where to go from here”, IEEE AUTOTESTCON, 2008, pp. 518-512, Septiembre 2008.

[82] “Test-System Development Guide. Choosing Your Test-System Software Architecture”, Nota de aplicación 1465-4, Agilent Technologies, 2004.

[83] Ding Lipei, Guo Xiaosong, Zhu Zhi, “Virtual Test System Based on Visual C++”, ICEMI '07. 8th International Conference on Electronic Measurement and Instruments, 2007, pp. 2-276 - 2-278, Julio-Agosto 2007.

[84] Winiecki, W., Karkowski, M., “A new Java-based software environment for distributed measuring systems design”, IEEE Transactions on Instrumentation and Measurement, Volumen 51, Número 6, pp. 1340-1346, Diciembre 2002.

[85] Bertocco, M., Cappellazzo, S., Carullo, A., Parvis, M., Vallan, A., “Virtual environment for fast development of distributed measurement applications”,

IEEE Transactions on Instrumentation and Measurement, Volumen 52, Número 3, pp. 681-685, Junio 2003.

[86] http://www.ni.com/mstudio/esa/

[87] Delle Femine, A., Gallo, D., Landi, C., Luiso, M., “Power-Quality Monitoring Instrument With FPGA Transducer Compensation”, IEEE Transactions on Instrumentation and Measurement, Volumen 58, Número 9, pp. 3149-3158, Septiembre 2009.

[88] Tlaczala, W., Grajner, G., Zaremba, M., “Virtual Laboratory With Simulated Nuclear Physics Experiments”, IEEE Transactions on Instrumentation and Measurement, Volumen 57, Número 8, pp. 1766-1770, Agosto 2008.

[89] Macii, D., Petri, D., “An Effective Power Consumption Measurement Procedure for Bluetooth Wireless Modules”, IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 4, pp. 1355-1364, Agosto 2007.

[90] Portillo, E., Cabanes, I., Marcos, M., Orive, D., Sanchez, J.A., “Design of a Virtual-Instrumentation System for a Machining Process”, IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 6, pp. 2616-2622, Dicimebre 2007.

[91] Nonclercq, A., Biest, A. V., De Cuyper, K., Leroy, E., Martinez, D. L., Robert, F., “Problem-Based Learning in Instrumentation: Synergism of Real and Virtual Modular Acquisition Chains”, IEEE Transactions on Education, Aceptado para future publicación, 2009.

[92] Tanmoy Maity, D. Ghosh, C.R. Mahata, “Effects of dielectric dispersion in multiplier chips”, Solid-State Electronics, Volumen 49, Número 10, pp. 1649-1654, Octubre 2005.

[93] Karl G. Wagner, James W. McGinity, “Influence of chloride ion exchange on the permeability and drug release of Eudragit RS 30 D films”, Journal of Controlled Release, Volumen 82, Números 2-3, pp. 385-39721, Agosto 2002.

[94] O. Körner, A. Van‘t Ooster, M. Hulsbos, “Design and performance of a measuring system for CO2 exchange of a greenhouse crop at different light levels”, Biosystems Engineering, Volumen 97, Número 2, pp. 219-228, Junio 2007.

[95] Measure Foundry: http://www.datatranslation.com/products/measurefoundry/

[96] Testpoint: http://www.mccdaq.com/testpoint.aspx

[97] Dasylab: http://www.dasylab.com/

[98] Ann L. Winblad. *Software orientado a objetos*. Ediciones Díaz de Santos, 1993.

[99] http://www.ni.com/LabVIEW/esa/

[100] AGILENT: http://www.home.agilent.com/agilent/product.jspx?nid=-34095.0.00&lc=spa&cc=ES

[101] Goldberg, H.; "What is Virtual Instrument?", Instrumentation & Measurement Magazine, IEEE", Volume 3, Issue 4, Dec. 2000 Page(s):10 - 13

[102] "Ingeniería del software", Ian Sommerville, traducción por María Isabel Alfonso Galipienso, Antonio Botia Martinez, 7ª Edición, Editorial Pearson Educación, 2005.

[103] Lovelock, C. Vandermerwe y Barbara Lewis, "Services Marketing". Englewood Cliffs, NJ: Prentice Hall, 1996.

[104] Ferrero, A., Salicone, S., Bonora, C., Parmigiani, M., "ReMLab: a Java-based remote, didactic measurement laboratory", IEEE Transactions on Instrumentation and Measurement, Volumen 52, Número 3, pp. 710-715, Junio 2003.

[105] Pianegiani, F.; Macii, D.; Carbone, P., "An open distributed measurement system based on an abstract client-server architecture", IEEE Transactions on Instrumentation and Measurement, Volumen 52, Número 3, pp. 686-692, Junio 2003.

[106] Andria, G.; Baccigalupi, A.; Borsic, M.; Carbone, A.P.; Daponte, P.; De Capua, C.; Ferrero, A.; Grimaldi, D.; Liccardo, A.; Locci, N.; Lanzolla, A.M.L.; Macii, D.; Muscas, C.; Peretto, L.; Petri, D.; Rapuano, S.; Riccio, M.; Salicone, S.; Stefani, F.; "Remote Didactic Laboratory "G. Savastano," The Italian Experience for E-Learning at the Technical Universities in the Field of Electrical and Electronic Measurement: Architecture and Optimization of the Communication Performance Based on Thin Client Technology", IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 4, pp. 1124-1134, Agosto 2007.

[107] Chi Chung Ko; Chen, B.M.; Shaoyan Hu; Ramakrishnan, V.; Chang Dong Cheng; Yuan Zhuang; Jianping Chen; "A web-based virtual laboratory on a frequency modulation experiment", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volumen 31, Número 3, pp. 295-303, Agosto 2001.

[108] Feng He; Gengquan Wang; Wei Li; Xiaolin Han; Jianhua Liu, "Object request brokers for distributed measurement", IEEE Computer Applications in Power, Volumen 14, Número 1, pp. 50-54, Enero 2001.

[109] Chen, J., Cui, H., "DOS Middleware Instrumentation for Ensuring Reproducibility of Testing Procedures", IEEE Transactions on Instrumentation and Measurement, Volumen 56, Número 1, pp. 56-62, Febrero 2007.

[110] Chen, S., "Open design of networked power quality monitoring systems", IEEE Transactions on Instrumentation and Measurement, Volumen 53, Número 2, pp. 597-601, Abril 2004.

[111] Daponte, P.; Di Penta, M.; Mercurio, G., “TransientMeter: a distributed measurement system for power quality monitoring”, IEEE Transactions on Power Delivery, Volumen 19, Número 2, pp. 456-463, Abril 2004.

[112] McDougall, P., “The power of peer-to-peer”, Information Week, 28 de Agosto de 2000, http://informationWeek.com.

[113] J. Newmarch, *Foundations of Jini 2 Programming*, USA: Apress, 2014.

[114] G. Alonso, F. Casati, H.Kuno y V. Machiraju, *Web Services. Concepts, Architectures and applications*, USA: Springer Science & Business Media, 2013

[115] I. Foster, C. Kesselman y otros, “Grid Services for distributed system integration”, IEEE Computer, 35(6), vol. 35, Número 6, pp. 37-46, 2002.

[116] A. Yazidi, H. Henao, G.A. Capolino, F. Betin, F. Filippetti, “A Web-Based Remote Laboratory for Monitoring and Diagnosis of AC Electrical Machines”, IEEE Transactions on Industrial Electronics, vol. 58, Número 10, pp. 4950 - 4959, 2011.

[117] A.P.J., Chandra y C. R. Venugopal, “Novel Design Solutions for Remote Access, Acquire and Control of Laboratory Experiments on DC Machines”, vol. 61, Número 2, pp. 349-357, 2012.

[118] Code Compose Studio Website: http://www.ti.com/tool/ccstudio

[119] S. Gallardo, F. Barrero, S.L. Toral, “Virtual instrumentation laboratory based on labVIEW. A case study: A DSPs course”, ICECE - International Conference On Engineering And Computer Education, Madrid, España, 2005.

[120] S. Gallardo, F. Barrero, S.L. Toral, M.J. Durán, “eDSPlab: A remote-accessed instrumentation laboratory for digital signal processors training based on the Internet”, IECON 2006. The 32th annual conference of IEEE industrial electronics society, Paris, Francia, 2006.

[121] M.R. Martínez-Torres, F. Barrero, S.L. Toral, S. Gallardo, “A digital signal processing teaching methodology using concept-mapping techniques”, IEEE Transactions on Education, vol. 48, pp. 422-429, 2005.

[122] S.L. Toral, F. Barrero, M.R: Martínez-Torres, S. Gallardo, M.J. Durán, “An electronic engineering curriculum design based on concept-mapping techniques”, International Journal of Technology and Design Education, vol. 17, pp. 341-356, 2007.

[123] S.L. Toral, M.R. Martínez-Torres, S. Gallardo, F. Barrero, “Análisis de una herramienta educativa remota sobre Procesadores Digitales de Señal desde la perspectiva de los Modelos de Aceptación Tecnológica”, PIXEL-BIT. REVISTA DE MEDIOS Y EDUCACION, vol. 29, pp. 87-100, 2007.

[124] S.L. Toral, M.R. Martínez-Torres, F. Barrero, S. Gallardo, “Desarrollo de un laboratorio remoto de procesadores digitales de señal basado en la familia TMS320C6000 y evaluación desde la perspectiva de los Modelos de

Aceptación Tecnológica", *Innovación y desarrollo de la calidad de la enseñanza universitaria - Experiencia de innovación universitaria, Instituto de Ciencias de la Educación*, *Universidad de Sevilla, vol. I, Número 17*, J.M. de Mesa, R. J. Castañeda, L. M. Villar. Editorial: Vicerrectorado de docencia: Sevilla-España, 2009, pp. 511-522.

[125] F. Barrero, S. l. Toral y M. Ruiz, *Procesadores Digitales de Señal de altas prestaciones*. Madrid-España: McGraw-Hill, 2005.

[126] Nasser Kehtarnavaz, Namjin Kim, *Digital Signal Processing System-Level Design Using LabVIEW*. USA: Elsiever, 2011.

[127] S. Gallardo, F. Barrero, S.L. Toral, "Estrategía EH-learning aplicada al aprendizaje de procesadores digitales de señal", Tecnologías Aplicadas a la Enseñanza de la Electrónica. TAEE 2008, Zaragoza, España, 2008.

[128] F. Barrero, S.L. Toral, M.R. Martínez-Torres, S. Gallardo, "Diseño y realización de un entorno WEB de ayuda a la impartición de clases practicas en una asignatura de tercer curso de la titulación de Ingeniería de Telecomunicación denominada "Complementos de Sistemas Electrónicos Digitales", *Innovación y desarrollo de la calidad de la enseñanza universitaria - Experiencia de innovación universitaria, Instituto de Ciencias de la Educación, Universidad de Sevilla*, vol. I, Número 12, J.M. de Mesa, R. J. Castañeda, L. M. Villar. Editorial: Vicerrectorado de docencia: Sevilla-España, 2007, pp. 439-452.

[129] A.J. Lillo, S. Gallardo, S.L. Toral, F. Barrero, "Laboratorio multimedia de procesamiento digital de señal usando en TMS320C3X DSP Starter Kit", Tecnologías Aplicadas a la Enseñanza de la Electrónica. TAEE 2004, Valencia, España, 2004.

[130] S. Gallardo, E. Molina, F. Barrero, S.L. Toral, M.J. Durán, "Aplicación de tecnologías multimedia para el aprendizaje asíncrono de instrumentación electrónica", Tecnologías Aplicadas a la Enseñanza de la Electrónica. TAEE 2006, Madrid, España, 2006.

[131] F. Barrero, S. Gallardo, A.J. Lillo, S.L. Toral, "Herramienta multimedia de ayuda en la impartición de un laboratorio de procesadores digitales de señal (DSPS)", PIXEL-BIT. REVISTA DE MEDIOS Y EDUCACION, vol.25, pp. 61-70, 2005.

[132] S.L. Toral, F. Barrero, M.R. Martínez-Torres, S. Gallardo, "Interactive Multimedia Teaching of Digital Signal Processors", Computer Applications in Engineering Education, vol. 15, pp. 88-98, 2007.

[133] S.L. Toral, M.R. Torres-Martínez, F. Barrero, S. Gallardo, "Realización de un libro electrónico multimedia sobre procesadores digitales de señal mediante Macromedia Director.", *Innovación y desarrollo de la calidad de la enseñanza universitaria - Experiencia de innovación universitaria, Instituto de Ciencias de la Educación, Universidad de Sevilla*, vol. I, Número 14, J.M. de Mesa, R. J. Castañeda, L. M. Villar. Editorial: Vicerrectorado de docencia: Sevilla-España, 2008, pp. 615-627.

[134] S. Gallardo, F. Barrero, S. L. Toral, “Building a Web-based virtual laboratory with VRML. A case study: An electronic instrumentation subject”, International Conference On Engineering And Computer Education. ICECE 2005, Madrid, España, 2005.

[135] S. Gallardo, F. Barrero, S.L. Toral, “Diseño de un entorno virtual de trabajo remoto basado en el modelado 3D de un laboratorio empleando VRML.”, *Innovación y desarrollo de la calidad de la enseñanza universitaria - Experiencia de innovación universitaria, Instituto de Ciencias de la Educación, Universidad de Sevilla*, vol. I, Número 14, J.M. de Mesa, R. J. Castañeda, L. M. Villar. Editorial: Vicerrectorado de docencia: Sevilla-España, 2008, pp. 535-550.

[136] S. Gallardo, J. Suardíaz, “End user quality of service measurement on Web based labs. A case study: eDSPlab”, III Congreso Iberoamericano sobre Calidad y Accesibilidad de la Formación Virtual - CAFVIR 2012, Madrid-España, 2012.

[137] S. Gallardo, F. Barrero y S.L. Toral, “Remote Instrumentation Laboratory for Digital Signal Processors Training”, *Practical Applications and Souction using LabVIEW Software*, S. Fole. Editorial InTech: Croatia, 2011, pp. 273-296.

[138] S. Gallardo, *Configuración de instalaciones domóticas y automáticas*, Madrid-España: Paraninfo, 2013.

[139] S. Gallardo, *Técnicas y procesos en instalaciones domóticas y automáticas*, Madrid-España: Paraninfo, 2013.

[140] L.M. Surhone, M.T. Tennoe, S.F. Henssonow, *Java Native Interface*, USA: VDM Publishing, 2010.

[141] S. Gallardo, F. Barrero, S.L. Toral, M.R. Martínez-Torres, “ReAL: Diseño de un laboratorio de instrumentación remoto y abierto basado en la API JNI de JAVA y el bus IEEE 488/GPIB”, *Innovación y desarrollo de la calidad de la enseñanza universitaria - Experiencia de innovación universitaria, Instituto de Ciencias de la Educación, Universidad de Sevilla*, vol. I, Número 14, J.M. de Mesa, R. J. Castañeda, L. M. Villar. Editorial: Vicerrectorado de docencia: Sevilla-España, 2009, pp. 391-399.

[142] N. Gunasekera, “Using the Scripting Utility in the Code Composer Studio - Integrated Development Environment”, Software Development Systems/Customer Support, Application Report, SPRA383A, Julio 2002.

[143] T. Christiansen, B. foy, L. Wall, J. Orwant, *Programming Perl: Unmatched power for text processing and scripting – 4th edition*, USA: "O'Reilly Media, Inc.", 2012.

[144] B. Aumaille, J2EE: *Desarrollo de aplicaciones Web*, Madrid-España: Ediciones ENI, 2002.

[145] J. Hunter, W. Crawford, *Java Servlet Programming*, USA: "O'Reilly Media, Inc.", 2001

[146] E. Condor, I. Soria, *Programación Web con CSS, JavaScript, PHP y AJAX*, Iván Soria Solís, 2014.

[147] F.J. Minera, *Ajax Web 2.0*, USERSHOP, 2007.

[148] Niderost, B., van de Giessen, M., Lourens, W., Krom, J., "The WebUmbrella Web-based access to distributed plasma-physics measurement data", IEEE Transactions on Nuclear Science, Volumen 49, Número 3, Parte 4, pp. 1579-1583, Junio 2002.

[149] Kalogeraki, V.; Fang Chen, "Managing distributed objects in peer-to-peer systems", IEEE Network, Volumen 18, Número 1, pp. 22-29, Enero-Febrero 2004.

[150] RiNGRID: www.ringrid.eu

[151] "State of the Art About Remote Laboratories Paradigms – Foundations of Ongoing Mutations", C. Gravier, J. Fayolle, B. Bayard, M. Ates and J. Lardon, International Journal of Online Engineering, Vol. 4, Issue 1, February 2008.

[152] N. Kehtarnavaz y N. Kim, *Digital Signal Processing System-Level Design using LabVIEW*, University of Texas at Dallas: Elsevier, 2005

[153] "Real-Time Digital Signal Processing in the Undergraduate Curriculum", Abdel-Qader, I. M., Bazuin, B. J., Mousavinezhad, H. S., and Patrick, J. K. .IEEE Transactions on Education, Vol. 46, No. 1, 2003, pp. 95-101, ISSN 0018-9359. 2003.

[154] Ministerio de Industria, Energía y Turismo: http://www.minetur.gob.es/telecomunicaciones/es-ES/Servicios/CalidadServicio/Paginas/Calidad.aspx

[155] International Standard Organization: www.iso.org

[156] « Recomendación UIT-T E.800» , Unión Internacional de Telecomunicaciones, 2008.

[157] "RFC 2386, A Framework for QoS-based Routing in the Internet", E. Crawley, R. Nair, B. Rajagopalan, H. Sandick, 1998, http://www.ietf.org/rfc/rfc2386.txt

[158] "Servicios Avanzados de Telecomunicación", María Carmen España Boquera, 2003, Editorial Diaz de Santos, ISBN: 84-7978-607-8.

[159] "QoS over Hetereogeneous Networks", Mario Marchese, 2007, Editorial John Wiley & Sons, Ltd., ISBN: 978-0-470-01752-4 (HB)

[160] "Recomendación UIT-T P.800 - SERIE P: CALIDAD DE TRANSMISIÓN TELEFÓNICA - Métodos de determinación subjetiva de la calidad de transmisión". ITU-T. 1996.

[161] S. Gallardo, F. Barrero y S.L. Toral, “Remote Instrumentation Laboratory for Digital Signal Processors Training”, *Practical Applications and Souction using LabVIEW Software*, S. Fole. Editorial InTech: Croatia, 2011, pp. 273-296.

[162] “Implementation of a web-based educational tool for digital signal processing teaching using the technological acceptance model”, Toral, S.L., Barrero, F., Martínez-Torres, M. R., Gallardo, S., Lillo, A. J., IEEE Transaction on Education, Vol. 48, Nº 3, August 2005, pp. 632-641, ISSN 0018-9359.

[163] “Perceived usefulness, perceived easy of use and user acceptance of information technology”, F.D. Davis, MIS Quartely, vol. 13, número 3, 1989.

[164] “A hybrid technology acceptance approach for exploring e-CRM adoption in organizations”, I.L. Wu, K.L. Wu, Behaviour & Information Technology, vol. 24, número 4, 2005.

[165] “A second generarion of multivariable analisys, Methods:”, C. Fornell, Praeger MY: Publishers, 1982.

[166] “The partial Least squares approach for structural equation modelling”, W.W., Chin, G.A. Marconlidez Ed. Modern Methods for business research, Mahwah, NJ, Erlbaum, pp. 295-336. 1998.

A. Anexos

A.1. Acrónimos

AJAX	Asynchronous JavaScript And XML
ASCII	American Standard Code for Information Interchange
ATE	Automated Test Equipment
CAN	Controller Area Network
CGI	Common Gateway Interface
CMS	Content Management System
DAQ	Data Acquisition
DEE	Dynamic Data Exchange
DOM	Document Object Model
DSP	Digital Signal Processor
DUT	Device under test
DHCP	Dynamic Host Configuration Protocol
FIFO	First In First Out
GPIB	General Purpose Instrumentation Bus
HP-IB	Hewlett-Packard Instrumentation Bus
HTML	HyperText Modelling Language
HTTP	HipertText Transfer Protocol
IVI	Interchangeable Virtual Instruments
LMS	Learning Management System
LXI	LAN extensions for Instruments
MOS	Mean Opinion Score
MXI	Multisystem Instrument Interface

P2P	Peer to Peer
PC	Personal Computer
POO	Programación orienta a objetos
PXI	PCI eXtension For Instrumentation
QoS	Quality of Service
RLS	Real Laboratory Server
RTDX	Real Time Data Exchange
SCPI	Standard Commands for Programable Instrumentation
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
TAD	Tarjetas de adquisición de datos
TAM	Technological Acceptation Model
UDDI	Universal Description, Discovery and Integration
VEE	Visual Engineering Environment
VISA	Virtual Instrumentation Systems Architecture
VLS	Virtual Laboratory Server
VRML	Virtual Reality Modelling Language
VXI	VME Bus eXtension for Instrumentation
WSDL	Web Services Description Language
WIMP	Windows, icons, and pull-down menus
XML	eXtensible Markup Language